Umsetzung von Augmented Reality in Rich Internet Applications am Beispiel von Adobe Flash

Autor:

Thomas Hurtig

- Bachelorarbeit -

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES (FH)

Fachbereich Medien
Studiengang Medientechnik

Mittweida, 9. August 2010

Umsetzung von Augmented Reality in Rich Internet Applications am Beispiel von Adobe Flash

Autor:

Thomas Hurtig

- eingereicht als Bachelorarbeit -

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES (FH)

Fachbereich Medien
Studiengang Medientechnik

Erstprüfer:	Prof. Dr.-Ing. Robert J. Wierzbicki
Zweitprüfer:	Dipl.-Ing. Sieglinde Klimant
Datum	9. August 2010
Ort:	Mittweida

Vorwort

Alle Beispielanwendungen wurden mit *Adobe Flash Builder 4* erstellt und liegen auf einer CD der Arbeit bei. Die Quelldateien befinden sich im jeweiligen Unterordner *src* und die kompilierte Datei im *swf*-Format im Unterordner *bin-release*.

Mein Dank für die Anregungen zur Themenfindung gebührt den Betreuern dieser Arbeit Prof. Dr.-Ing. Robert J. Wierzbicki und Dipl.-Ing. Sieglinde Klimant.

Bibliographische Beschreibung

Hurtig, Thomas:
Umsetzung von Augmented Reality in Rich Internet Applications am Beispiel von Adobe Flash. - 2010. - 61 S.
Mittweida, Hochschule Mittweida, Fakultät Medien, Bachelorarbeit, 2010

Referat

Ziel dieser Arbeit ist es, die Umsetzung von Augmented Reality Anwendungen als Rich Internet Application aufzuzeigen. Dafür werden die für die Erstellung einer solchen Anwendung grundlegenden Themengebiete der Displayarten, der Kalibrierung und Registrierung sowie der Darstellung von virtuellen Objekten in einer realen Umgebung erläutert und anschließend ausgewählte Verfahren am Beispiel von *Adobe Flash* umgesetzt. Dabei wird besonders auf verschiedene markerbasierte Registrierungsverfahren und auf die Darstellung von dreidimensionalen Objekten in *Flash* eingegangen sowie eine Bewertung der vorgestellten Systeme vorgenommen. Es wird unter zu Hilfenahme zweier Tracking-Bibliotheken das Konzept für eine markerlose Augmented Reality Anwendung erstellt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

EWK-Continuum	Extent of World Knowledge Continuum; dt.: Kenntnisstand über die Welt
HAD	Head Attached Display; dt.: Am Kopf befestigter Bildschirm
HMD	Head Mounted Display; wörtlich: Am Kopf montiertet Bildschirm
RIA	Rich-Internet-Application; dt.: Reichhaltige Internetanwendung
RV-Continuum	Reality-Virtuality Continuum; dt.: Realitäts-Virtualitäts-Konitnuum
WOW	Window on the World; dt.: Fenster zur Welt

1. Einleitung

Es war schon immer eine faszinierende Vorstellung, unsere reale Welt mit künstlichen, computergenerierten Inhalten zu erweitern. Diese Art von Anwendungen und Systemen nennt sich Augmented Reality. Auf deutsch könnte man dies mit angereicherter oder erweiterter Realität übersetzen.

Ein Blick in das Science-Fiction Genre zeigt, wie zukünfte Anwendungen auf diesem Gebiet aussehen können. Sei es das medizinische Programm auf der *Enterprise* aus der *Start Trek* Reihe, das einen virtuellen Arzt scheinbar zum Leben erweckt oder die Übermittlung einer Nachricht mittels einer dreidimensionalen Projektion. Auch wenn diese Anwendungen noch in das Reich der Fantasie gehören, könnten sie doch bald in ähnlicher Form Realität werden. In vielen Medien begegnet uns bereits heute Augmented Reality, die aber oft gar nicht als solche wahrgenommen wird. Hierbei handelt es sich meist um visuelle Erweiterungen eines realen Bildes. Von der Fernsehübertragung bis hin zu Computerspielen wie dem *EyePet* von Sony oder mobilen Anwendungen wie etwa dem *Wikitude Drive* sind die Einsatzgebiete von Augmented Reality vielfältig. Aber auch in der Medizin, in der Produktentwicklung oder im militärischen Bereich finden sich weitere Einsatzmöglichkeiten.

Die Entwicklung von Augmented Reality reicht bis zu den Anfängen der Computertechnik zurück. Bereits 1968 wurde von Ivan Sutherland an der Havard Universität der erste funktionsfähige Prototyp eines Augmented Reality Systems entwickelt.[1] Zur damaligen Zeit war jedoch ein enormer technischer Aufwand nötig, um solche Anwendungen umsetzen zu können. Durch immer leistungsfähigere Rechner ist es jetzt möglich, Augmented Reality Anwendungen auf Consumer Hardware auszuführen. Hierbei bietet vor allem das Internet aber auch Smartphones eine geeignete Plattform, um bereits heute Augmented Reality für den Endverbraucher erlebbar zu machen. In Zukunft werden Augmented Reality Anwendungen immer mehr den Weg in unser alltägliches Leben finden.

[1] vgl. SUTHERLAND 1968, 757 ff.

1.1. Zielsetzung

Ziel dieser Arbeit ist es, die Umsetzungsmöglichkeiten von Augmented Reality in Rich-Internet-Applications (RIAs) aufzuzeigen. Dazu werden die verschiedenen technischen Ansätze von Augmented Reality vorgestellt und deren Umsetzung in RIAs erläutert. Weiterhin sollen in dieser Arbeit Probleme deutlich werden, die bei der Entwicklung von Augmented Reality Anwendungen im Allgemeinen und im Kontext der Entwicklung einer Augmented Reality RIA entstehen. Da Augmented Reality Anwendungen sich häufig auf die Erweiterung der Realität durch visuelle Elemente beschränken, soll das auch der Schwerpunkt dieser Arbeit sein.

In Kapitel 2 wird der Begriff Augmented Reality erläutert und mit verwandten Bereichen wie der Augmented Virtuality verglichen und in das Reality-Virtuality Continuum (RV-Continuum) nach Milgram eingeordnet. Der Autor zeigt eine Auswahl an möglichen Typisierungen von Augmented Reality Anwendungen anhand ihrer technischen Umsetzung. Weiter wird auf das Thema der Kalibrierung und Registrierung und auf auftretende wahrnehmungsspezifische Probleme eingegangen.

Kapitel 3 geht auf die Umsetzungsmöglichkeiten von Augmented Reality in RIAs am Beispiel von *Adobe Flash* ein. Ein besonderer Schwerpunkt liegt dabei auf ausgewählten Bibliotheken für *Flash*, die die Umsetzung von Augmented Reality ermöglichen bzw. erheblich vereinfachen. Diese werden anhand von ausgewählten Kriterien verglichen und auf ihre Einsatzmöglichkeiten in Bezug auf Augmented Reality Anwendungen hin untersucht.

In Kapitel 4 wird das Konzept für eine Augmented Reality Anwendung auf Basis von *Adobe Flash* beschrieben. In dieser Anwendung bekommt der Benutzer die Möglichkeit, eine virtuelle Brillenanprobe am heimischen Rechner durchführen zu können. Dazu wird über das Videobild einer Webcam ein 3D-Modell einer Brille auf den Kopf des Nutzers *gelegt*. Um einen möglichst realistischen Eindruck zu vermitteln, soll diese in Ausrichtung und Position der Kopfbewegung des Benutzers folgen. Die Problemstellung hierbei ist, eine geeignete Tracking- und Registrierungs-Methode zu finden. Weiter müssen Verdeckungseffekte des Kopfes gegenüber dem 3D-Modell berücksichtigt werden. Da das Tracking und die Erstellung von 3D-Modellen nicht Gegenstand dieser Arbeit ist, werde ich auf diese Themenbereiche nur insofern ein-

gehen, als das die von aufgezeigten Techniken in ihren Grundzügen erklärt werden. Ebenfalls nicht Bestandteil ist die Erstellung bzw. Umsetzung dieser Anwendung.

2. Augmented Reality

Augmented Reality beschreibt die Ergänzung bzw. Anreicherung der Realität durch virtuelle Elemente. Dabei wird die Wahrnehmung der realen Welt durch künstliche Sinnesreize erweitert. Der Benutzer bekommt Informationen, die er allein mit seinen Sinnen nicht wahrnehmen könnte.[2] Die Sinne des Benutzers werden also in gewisser Weise erweitert. In visuellen Augmented Reality Anwendungen wird ein Abbild der Realität wie etwa ein Videobild mit computergenerierten Grafiken kombiniert. Die bevorzugte Entwicklung von visuellen Augmented Reality Systemen liegt zumeist an der einfacheren technischen Umsetzbarkeit gegenüber anderen Sinneswahrnehmungen wie zum Beispiel dem Hör- oder Geruchssinn. Dennoch beschränkt sich Augmented Reality nicht darauf, sondern kann all unsere Sinne ansprechen. Es können ebenfalls Objekte aus einer realen Szene entfernt werden, zum Beispiel um zu sehen, was sich hinter einer Wand verbirgt.

2.1. Definition

2.1.1. Einordnung nach Milgram

Um den Begriff Augmented Reality besser einordnen zu können, muss man die verwandten Bereiche Augmented Virtuality, Virtual Reality und den Begriff der Realität betrachten. Dazu hat Milgram im *Reality-Virtuality Continuum* diese Begriffe zueinander in Beziehung gesetzt. (Abb. 1)

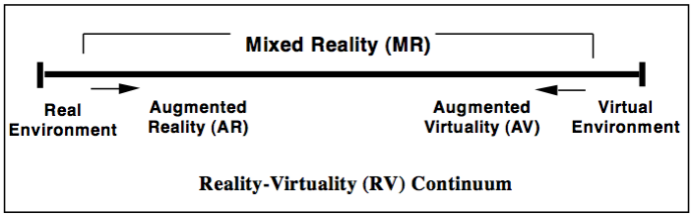


Abbildung 1.: Reality-Virtuality Continuum nach Milgram
(Abb. aus: MILGRAM/TAKEMURA/UTSUMI et al. 1994, 283)

[2] vgl. AZUMA 1997, 3

In der virtuellen Realität wird der Benutzer komplett von einer künstlichen Welt umgeben. Der Unterschied zwischen künstlicher und realer Welt wird von Milgram dabei anhand der Gültigkeit von Naturgesetze festgelegt. Die Virtual Reality muss nicht zwangsweise an physikalische Gesetze wie zum Beispiel Zeit oder Schwerkraft gebunden sein. Die Realität hingegen kann sich dieser Gesetze nicht entziehen.[3] Weiter wird jegliche Art von abgetasteten Bilddaten als real angesehen, bei denen der Computer keinerlei Kenntnisse über den inhaltlichen Kontext des Bildes hat.[4] Im RV-Continuum wird deutlich, dass die Übergänge zwischen Augmented Reality und Augmented Virtuality fließend sind. Einzig die Realität und die Virtuelle Realität sind klar von den übrigen Bereichen abgegrenzt. Zusammenfassend ist der gesamte Bereich zwischen diesen beiden Polen als Mixed Reality bezeichnet.

Um ein System als Augmented Reality oder Virtual Reality einzuordnen, genügt es nicht, die virtuellen gegen die realen Elemente in einer Szene quantitativ gegeneinander abzuwiegen. Diese Vorgehensweise scheitert zum Beispiel dann, wenn eine komplett modellierte Welt mit reinen Bilddaten aus der realen Welt texturiert wird. Das Bild scheint auf dem ersten Blick real, da nur reale Elemente sichtbar sind. Dennoch handelt es sich dabei um Augmented Virtuality, da die gesamte Welt künstlich erstellt wurde.[5] Daraus lässt sich ableiten, dass man die einem System zu Grunde liegende Welt zur Einordnung heranziehen kann. Milgram und Colquhoun haben dazu ein *Extent of World Knowledge Continuum (EWK-Continuum)* entwickelt.

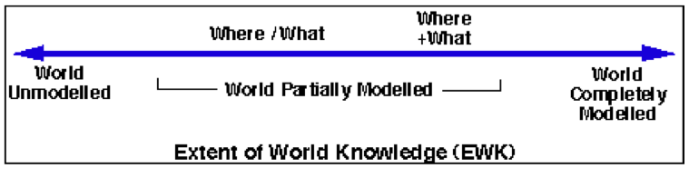


Abbildung 2.: Extent of World Knowledge Continuum nach Milgram
(Abb. aus: MILGRAM/COLQUHOUN JR. 1999, 7)

Der mittlere Teil des Continuums beschreibt die teilweise Modellierung der Welt. Es existiert somit ein Wissen über Objekte in der gezeigten Szene. Dies kann zum Beispiel die Position eines Objektes, seine geometrische Form oder seine Beziehung zu anderen Objekten sein. Je mehr Wissen über die Welt vorhanden ist, desto mehr ist sie modelliert. Wenn alle Informationen über alle Objekte vorhanden sind, somit die Welt komplett modelliert ist, spricht man von Virtual Reality.

[3] vgl. MILGRAM/TAKEMURA/UTSUMI et al. 1994, 283
[4] vgl. MILGRAM/COLQUHOUN JR. 1999, 7
[5] vgl. MILGRAM/COLQUHOUN JR. 1999, 12

2.1.2. Definition nach Azuma

Azuma definiert Augmented Reality in einem engeren Kontext anhand von technischen Details. Danach müssen Augmented Reality Systeme folgende Merkmale aufweisen:[6]

- Kombination von realen und virtuellen Elementen
- Interaktivität in Echtzeit
- Dreidimensionale Registrierung

Der erste Punkt deckt sich mit dem Bereich der Mixed Reality im RV-Continuum von Milgram. Ein neuer Aspekt ist die Interaktivität in Echtzeit und die dreidimensionale Registrierung. Durch diese Voraussetzungen werden im Gegensatz zur Definition von Milgram reine zweidimensionale Überblendungen ausgeschlossen. Weiter werden 3D-Animationen in Filmen nicht als Augmented Reality gesehen, da diese nicht interaktiv und in Echtzeit sind.[7]

2.1.3. Schlussfolgerung aus den genannten Definitionen

Wie im RV-Continuum von Milgram ist auch bei Azuma eine genauere Abgrenzung zwischen Augmented Reality und Augmented Virtuality nicht gegeben. Alle Kriterien können auch für Augmented Virtuality Systeme zutreffen. Eine klare Definition ist also nach objektiven Kriterien in der Praxis nur schwer möglich. Nach eigener Einschätzung ist eine Kombination aus den Ansätzen von Milgram und Azuma die beste Möglichkeit, Augmented Reality zu definieren und gegenüber Augmented Virtuality und Virtual Reality abzugrenzen.

[6] AZUMA 1997, 2
[7] vgl. AZUMA 1997, 2

2.2. Arten von Augmented Reality Displays

Es gibt verschiedene Möglichkeiten, Augmented Reality Systeme systematisch einzuordnen. Dabei liegt eine Einordnung anhand der technischen Umsetzung nahe. Die Literatur bezieht sich dabei auf die verschiedenen Displayarten zur Darstellung von Augmented Reality. Ein mögliches Schema ist, zwischen See Through und Monitor Based Augmented Reality zu unterscheiden.[8] Ein weiterer Ansatz ist, nach Body Attached und Spatial Displays zu trennen.[9] Cawood und Fiala unterscheiden hingegen zwischen Magic Mirror und Magic Lense.[10]

In dieser Arbeit soll eine Einteilung nach Bimber und Raskar erfolgen. Nach diesem Schema gibt es drei große Gruppen. Head Attached, Hand Held und Spatial Displays.[11] Abbildung 3 zeigt, an welcher Stelle bei den verschiedenen Displayarten das Augmented Reality Bild entsteht.

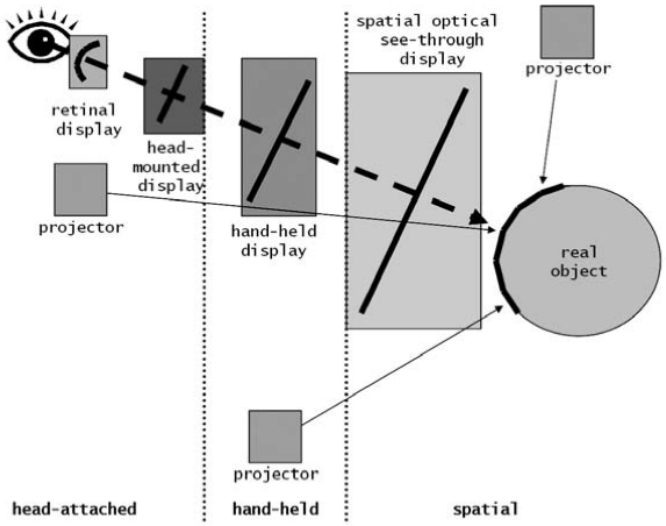


Abbildung 3.: Entstehung der Bilder bei den verschiedenen Augmented Reality Displays (Abb. aus: BIMBER/RASKAR 2005, 72)

[8] vgl. MILGRAM/TAKEMURA/UTSUMI et al. 1994, 284
[9] vgl. BIMBER/RASKAR 2005, 83
[10] vgl. CAWOOD/FIALA 2008, 3 f.
[11] vgl. BIMBER/RASKAR 2005, 91

2.2.1. Head Mounted Displays

Bei Head Mounted Displays (HMDs) bzw. Head Attached Displays (HADs) wird die Anreicherung direkt im Sichtfeld des Benutzers vorgenommen. Es erfolgt eine Unterscheidung in Video See Through HMDs und Optical See Through HMDs. Eine Spezialform des HMDs ist dabei das Netzhaut-Display, da hier die Informationen mittels einer Laseroptik direkt auf die Netzhaut projiziert werden. Der Vorteil von HMDs ist zum einen, dass der Benutzer beide Hände frei hat. Dies ist besonders bei industriellen, medizinischen und militärischen Anwendungen von Bedeutung. Zum anderen empfindet der Benutzer die virtuellen Elemente mehr als Teil seiner realen Wahrnehmung, da diese direkt im Sichtfeld und aus einem egozentrischen Blickwinkel dargestellt werden.

Ein Problem, das alle HMDs mit sich bringen können, ist die so genannte Simulator Sickness. Dabei bekommt der Benutzer physische Symptome wie Übelkeit, Schwindel oder Kopfschmerzen. Dies liegt daran, dass die Informationen die das Gehirn von Augen und Gleichgewichtsorgan bekommt, nicht zusammenpassen. Bei der Simulator Sickness senden die Augen ein Signal der Bewegung, während das Gleichgewichtsorgan Stillstand signalisiert. Diese widersprüchlichen Informationen bewirkt bei manchen Menschen die Eingangs erwähnten Erscheinungen. Dieser Effekt kann, vor allem bei großen Displayanordnungen, auch bei Spatial Displays (siehe 2.2.3) auftreten.[12,13]

Bei Video See Through HMDs wird das Umgebungsbild mittels einer Kamera aufgenommen, an einen Computer gegeben und dort mit virtuellen Elementen erweitert. Anschließend wird das erweiterte Videobild auf kleinen Bildschirmen, die direkt vor den Augen liegen, wiedergegeben.

Bei Optical See Through HMDs hingegen sieht der Nutzer durch eine Art halbdurchsichtiges Display auf die reale Welt. Die virtuellen Elemente werden dabei auf diesem Display abgebildet. Dies geschieht entweder durch eine Projektion auf eine Art durchsichtigen Spiegel oder durch transparente LCDs oder auch OLEDs. Somit entsteht der Eindruck einer Überlagerung von realen und virtuellen Elementen. [14,15,16] Die Optical See Through HMDs benötigen eine sehr präzise Kopf- und

[12] vgl. SCHMIDT/SCHMITZ
[13] vgl. BIMBER/RASKAR 2005, 75
[14] vgl. BIMBER/RASKAR 2005, 74 f.
[15] vgl. MILGRAM/TAKEMURA/UTSUMI et al. 1994, 284
[16] AZUMA 1997, 10

Körpertrackingtechnik, damit die überlagerten virtuellen Objekte an der korrekten Position dargestellt werden. Auch das gesamte HMD muss deshalb exakt kalibriert werden. (siehe 2.3) Kleinste Veränderungen würden zu einer inkorrekten Darstellung führen. Dieser Umstand ist bedeutsamer, wenn eine stereoskopische Darstellung erreicht werden soll.[17,18] Durch automatische Rekalibrierung des Systems, zum Beispiel anhand von Augentracking, kann dieses Problem minimiert werden. Die virtuellen Elemente werden in einer bestimmten Tiefenebene im Bild dargestellt. Dadurch ist der Nutzer gezwungen, zwischen den beiden Fokuspunkten des realen und virtuellen Inhalts zu wechseln. Das ist nur der Fall, wenn sich die beiden Fokuspunkte unterscheiden. Dieses Problem wird auch als *fixed focal length problem* bezeichnet.[19,20] Eine Lösung wäre hier ein Verfahren, das den Fokuspunkt des Auges ermittelt und anhand dieser Information die Platzierung der virtuellen Elemente entsprechend anpasst. Das Problem der festen Brennweite ist bei Video See Through HMDs nicht vorhanden, da das Auge nur die Bildschirme fokusiert und somit seinen Fokus nicht ändert. Weitere von Bimber und Raskar angeführte Probleme dieser Displayarten wie zum Beispiel die geringe Auflösung und die großen und unergonomischen Vorrichtungen sind heute zum Teil schon gelöst.

Beim Netzhaut- bzw. Retinal-Display wird das virtuelle Objekt mittels eines Lasers direkt auf die Netzhaut des Benutzers projiziert. (Abb. 4) Der große Vorteil dieser Technologie liegt darin, dass kein Bildschirm vor dem Auge platziert werden muss. Dadurch ist das Sichtfeld des Benutzers nicht wesentlich eingeschränkt. Die projizierten Objekte können mit einer höheren Auflösung und Schärfe dargestellt werden, als es mit Bildschirmtechnologien wie Video See Through und Optical See Through möglich ist. Weitere Vorteile sind der geringe Stromverbrauch und die im Vergleich zu den anderen Displayarten kleinere Bauweise.[21] Dadurch können diese Displays auch als Aufsatz für eine normale Brille konstruiert werden.

[17] vgl. MILGRAM/TAKEMURA/UTSUMI et al. 1994, 284
[18] vgl. BIMBER/RASKAR 2005, 75
[19] vgl. BIMBER/RASKAR 2005, 75
[20] vgl. AZUMA 1997, 16
[21] vgl. BIMBER/RASKAR 2005, 73 f.

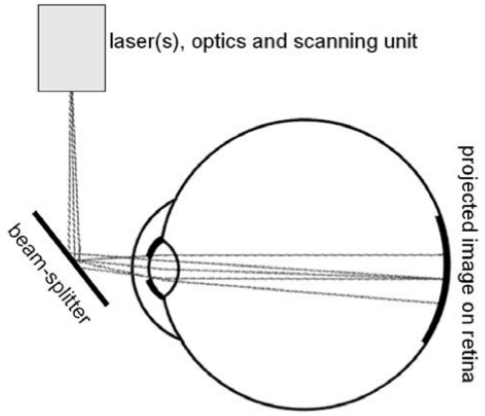


Abbildung 4.: Schematische Darstellung eines Netzhautdisplays
(Abb. aus: BIMBER/RASKAR 2005, 73)

2.2.2. Handheld Displays

Heutige Mobiltelefone und PDAs bieten nahezu alle die geeignete Hardware, um Augmented Reality Anwendungen zu realisieren. Vor allem Anwendungen auf dem Konzept der *Window on the World (WOW)* Metapher (siehe 2.2.3) sind mit diesen Geräten gut umsetzbar, da fast alle Geräte über eine integrierte Kamera und ein Display verfügen. Mit zusätzlichen Informationen zu dem Standort des Benutzers mittels GPS Daten zeigen sich weitere Möglichkeiten für Augmented Reality auf. Alle Stufen einer Augmented Reality Anwendung, von der Aufnahme des realen Bildes, die Verarbeitung, Analyse und Anreicherung der Bildaten sowie deren Ausgabe sind in einem Gerät vereint. Wie bei den anderen Displayarten auch, gibt es bei den Handheld Displays neben den Video See Through Displays auch Optical See Through Displays. Die Funktionsweise ist analog zu den bereits im Abschnitt 2.2.1 bzw. 2.2.3 beschriebenen Displayarten.

2.2.3. Spatial Displays

Unter Spatial Displays lassen sich alle Displayarten zusammenfassen, die nicht am Körper des Benutzers getragen, sondern im Raum platziert werden. Gegenüber HMDs kann ein vollständiges *Eintauchen* des Benutzers in die Szene nur teilweise stattfinden. Lediglich durch sehr große Displayanordnungen kann der Betrachter das Gefühl

erhalten, komplett in der Augmented Reality Welt zu sein. Wie auch bei den HMDs wird bei den Spatial Displays zwischen Video See Through und Optical See Through unterschieden. Weiter gibt es die sogenannte Direct bzw. Projection Based Augmentation.

Bei Screen-Based Video See Through Display wird wie bei den Video See Through HMDs die Umgebung durch ein Live-Video auf dem Bildschirm abgebildet. Der Betrachter blickt durch den Bildschirm auf die reale Welt. Hierbei wird jedoch die Bezeichnung See-Through nicht wörtlich genommen, da der Bildschirm sowohl die vor ihm als auch die hinter ihm liegende Szene darstellen kann. Diese beiden Sichtweisen werden auch als Magic Mirror und Magic Lense bezeichnet. Der Magic Mirror zeigt wie ein Spiegel die vor ihm liegende Szene, etwa mittels Live-Videobild und erweitert diese mit virtuellen Elementen. Unter Magic Lense werden alle Systeme zusammengefasst, bei denen der Benutzer *durch* ein Display auf die reale Welt schaut.[22] Letzteres wird auch als *WOW* bezeichnet.[23,24] Ein wesentlicher Vorteil dieser Displayart ist die kostengünstige Umsetzung, da keine spezielle Hardware erforderlich ist. Nachteilig ist, dass je nach Monitorgröße das Sichtfeld eingeschränkt ist.[25]

Unter Spatial Optical See Through Displays versteht man Systeme, die an einer festen Position im realen Raum stehen. Hierunter fallen zum Beispiel transparente oder holographische Bildschirme. Die holographischen Systeme arbeiten meist mit einer speziellen Kombination aus Spiegel und Optiken, um den Eindruck eines dreidimensionalen Bildes zu erzeugen.

Bei der Projection Based Augmented Reality wird die Erweiterung der realen Welt für den Betrachter unmittelbar sichtbar. Es werden keine künstlichen Vermittler zwischen Auge und der Augmented Reality Szene benötigt. Aus diesem Grund nennt man diese Art von Augmented Reality auch *Direct Augmentation*, da die Umwelt direkt beeinflusst wird. Dazu werden per Frontprojektion Informationen direkt auf die realen Objekte projiziert. Ein Probleme ist hierbei, dass der Benutzer oder andere reale Objekte die Projektion verdecken können. Weiter sind herkömmliche Projektoren

[22] vgl. CAWOOD/FIALA 2008, 3 f.
[23] vgl. BIMBER/RASKAR 2005, 84
[24] vgl. MILGRAM/TAKEMURA/UTSUMI et al. 1994, 284
[25] vgl. BIMBER/RASKAR 2005, 84 f.

auf eine bestimmte Fokusebene beschränkt. Bei der Projektion auf unebene Flächen kann dadurch das Bild unscharf oder verzerrt dargestellt werden.[26]

2.3. Kalibrierung und Registrierung

Durch Kalibrierung und Registrierung eines Augmented Reality Systems wird der Bezug zwischen Realität und den virtuellen Elementen hergestellt. Dies ist nötig, um die virtuellen Objekte korrekt in der realen Szene darstellen zu können. Die Übergänge zwischen den beiden Verfahren sind dabei fließend. McGarrity definiert die Kalibrierung als

> *„...the process of instantiating parameter values for [mathematical] „models“which map the physical environment to internal representations, so that the computer's internal model matches the physical world.“* [27]

Es müssen also zuerst alle nötigen Parameter der Komponenten eines Augmented Reality Sytems gefunden werden, die Einfluss auf die dargestellte, erweiterte Szene haben. Diese Parameter betreffen hauptsächlich die Beziehungen des Systems Kamera-Benutzer-Display. Zum Beispiel muss bei HMDs der Versatz zwischen dem Blickwinkel der Kamera und dem des Benutzers berücksichtigt werden. Zusätzlich müssen die Beziehungen zwischen HMD und Auge beachtet werden, da ein Verrutschen hier eine Veränderung der Position der virtuellen Elemente zur Folge hat.[28,29] Außerdem sind die Kameraparameter wie die Verzerrung durch das Objektiv und die Brennweite wichtige Faktoren, die das Funktionieren eines Augmented Reality Systems beeinflussen können.[30]

Registrierung ist das Verfahren, das die korrekte Platzierung von virtuellen Objekten in der realen Szene gewährleistet. Dabei geht es um die Position, Ausrichtung und Größe der virtuellen Elemente in Bezug auf die realen Objekte. Damit die Darstellung in Position und Perspektive korrekt ist, muss die Position der virtuellen Kamera

[26] vgl. BIMBER/RASKAR 2005, 87 ff.
[27] MCGARRITY/TUCERYAN 1999, 2
[28] vgl. MCGARRITY/TUCERYAN 1999, 1 ff.
[29] vgl. KATO/BILLINGHURST 1999, 4
[30] Camera Calibration. `http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm`, letzter Aufruf: 07.06.2010

mit der der realen Kamera übereinstimmen. Das primäre Ziel der Registrierung ist, eine geeignete Transformation zwischen Punkten der realen und der virtuellen Welt zu finden. Optical See Through Displays müssen hier wieder die Beziehungen im System Kamera-HMD-Augen beachten. Video See Through Displays benötigen lediglich die Transformation zwischen den 3D-Koordinaten der realen Welt und den 2D-Koordinaten des Displays.[31] In der Literatur wird die Registrierung bei Augmented Reality auch als Tracking bezeichnet, da die Berechnung der Transformation permanent geschieht und meist anhand von realen Objekten erfolgt.

Prinzipiell kann man zwischen markerbasierter und markerloser Registrierung unterscheiden. Mit markerbasiert sind dabei alle Verfahren gemeint, bei denen der realen Welt physikalisch sogenannte Marker hinzugefügt werden. Dies kann in Form eines gedruckten Musters auf einem Blatt Papier sein, aber auch das Benutzen von LEDs, Infratot-Systemen oder magnetischen Systemen als Bezugssystem fallen in diese Kategorie. Bei markerlosen Systemen werden der realen Welt physikalisch keine Marker hinzugefügt. Die markerlosen Systeme arbeiten meist auf der Basis von Bildanalyse. Dabei wird das Videobild auf Merkmale untersucht. Diese Merkmale können zum Beispiel bestimmte Farben, Muster oder geometrische Formen sein, die Auskunft über Position, Ausrichtung oder Art der enthaltenen Objekte geben. Anhand dieser Informationen lassen sich Teile der realen Szene modellieren und somit Rückschlüsse für eine korrekte Registrierung der virtuellen Objekte ziehen.

Die korrekte Kalibrierung und Registrierung ist für Augmented Reality Anwendungen unerlässlich. Bei medizinischen Augmented Reality Anwendungen zum Beispiel entscheiden bereits einige Millimeter Abweichung über die Einsetzbarkeit eines Systems. Die korrekte Kalibrierung und Registrierung ist daher eines der größten Probleme von Augmented Reality.[32]

2.3.1. Markerbasierte Registrierung

Die markerbasierte Registrierung wird bei vielen Augmented Reality Anwendungen eingesetzt. Das System bekommt durch den Marker Informationen, anhand denen es die virtuellen Elemente in Größe, Position und Ausrichtung korrekt platzieren kann. Es gibt verschiedene Arten von Markern wie Pattern-Marker, LEDs oder auch

[31] vgl. KATO/BILLINGHURST 1999, 4
[32] vgl. AZUMA 1997, 18

Infrarot-Systeme wie sie beispielsweise bei dem *Nintendo Wii Controller* eingesetzt werden. Im anschließenden Kapitel soll auf die quadratischen Pattern-Marker eingegangen werden, da diese in vielen Augmented Reality Anwendungen zum Einsatz kommen.

2.3.1.1. Quadratische Pattern-Marker

Bei dieser Methode werden in der realen Umgebung Marker (Abb. 5) angebracht. Die Software durchsucht das Videobild nach diesen Markern. Dabei ist dem System das Aussehen bzw. der Inhalt der Marker bekannt. Dadurch kann es die verschiedenen Marker im Bild erkennen und auch voneinander unterscheiden. Durch die Position und Ausrichtung der Marker kann die Software eine geeignete Transformationsmatrix von realer zu virtueller Kamera erstellen. Anhand dieser können dann die virtuellen Objekte in korrekter Position und Ausrichtung im Bild platziert werden.

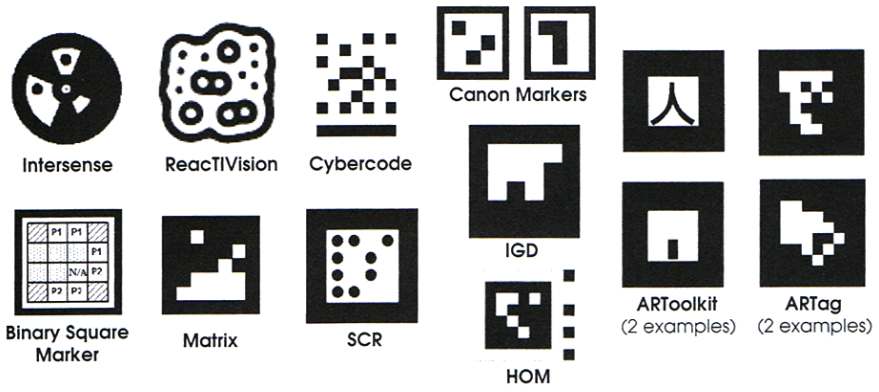


Abbildung 5.: Verschiedene Pattern-Marker Systeme
(Abb. aus: CAWOOD/FIALA 2008, 286)

Eines der verbreitesten Systeme der quadratischen Patter-Marker ist *ARToolKit*. Das ist eine C/C++ Bibliothek, mit der sich markerbasierte Augmented Reality Anwendungen umsetzen lassen. Es werden Funktionen für das Tracking der Marker und zur Errechnung der Kameratransformation bereitgestellt. Das Rendern von 3D-Modellen oder sonstige für Augmented Reality Anwendungen notwendige Funktionen sind nicht Bestandteil dieser Bibliothek. *ARToolKit* ist unter der General Public License für den nicht kommerziellen Gebrauch frei einsetzbar. Ein Vorteil dieser Bibliothek ist, dass sie bereits für viele andere Plattformen portiert wurde. So gibt es die

ARToolKit Bibliothek unter anderem für *Java (NyARToolKit)*, *Adobe Flash (FLARToolKit)*, *Microsoft Silverlight (SLARToolKit)* und für das Betriebssystem *Android (AndAR)*. Für *ARToolKit* und deren Portierungen sind auch kommerzielle Lizenzen erhältlich.[33] Eine Gegenüberstellung weiterer Systeme sowie die jeweiligen Vor- und Nachteile sind in der Arbeit von Daniel Abawi[34] zu finden.

2.3.1.2. Funktionsweise am Beispiel von ARToolKit

ARToolKit arbeitet wie die meisten anderen Markersysteme mit einer Reihe von Bildverarbeitungsprozessen wie Schwellwertoperationen und Kantendetektion. Die verwendeten Marker müssen einen möglichst hohen Kontrast im Vergleich zum restlichen Bild aufweisen. Daraus resultieren die schwarz-weiß Marker (Abb. 5), da hier der größtmögliche Kontrast gegeben ist.

Die Kalibrierung der Kamera ist in *ARToolKit* nicht zwingend notwendig, da eine Art Standardkonfiguration der Kameraparameter in der Bibliothek enthalten ist. Mit dieser Standardkonfiguration lassen sich bereits gute Ergebnisse erzielen. Der Vorteil dieser Standardkonfiguration ist, dass bei einem Systemwechsel nicht zwingend eine manuelle Rekalibrierung der Kamera notwendig ist. Dennoch bietet *ARToolKit* auch die Möglichkeit, die individuellen Kameraparameter zu ermitteln und eine kameraspezifische Parameterdatei anzulegen. Die Erstellung der Parameterdatei gliedert sich in die Ermittlung der Objektiverzerrung und in die Ermittlung der Brennweite der Kamera. In beiden Fällen wird mit einem vordefinierten Kalibrierungsmuster gearbeitet. Der Nutzer muss dabei jeweils manuell bestimmte Bereiche markieren (Abb. 6) und virtuelle Linien mit denen des Kalibrierungsmuster in Übereinstimmung bringen. (Abb. 7) Die Vorgänge werden jeweils mit unterschiedlichen Abständen der Kalibrierungsmuster zur Kamera mehrmals wiederholt. Die Software errechnet dann anhand der markierten Einzelbilder jeweils die Objektivverzerrung und die Brennweite der Kamera. Aus diesen Daten wird anschließend eine für die verwendete Kamera spezifische Konfigurationsdatei erstellt. Die Kalibrierungsprozedur ist nicht zwingend erforderlich, kann aber zu besseren Ergebnissen bei der Erkennung und Verfolgung der Marker führen.[35]

[33] siehe `http://www.artoolworks.com`

[34] ABAWI 2005 a

[35] vgl. Camera Calibration. `http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm`, letzter Aufruf: 07.06.2010

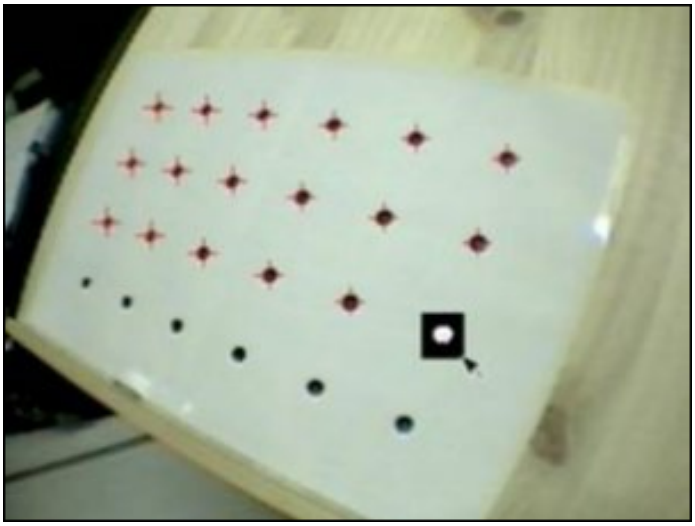


Abbildung 6.: Ermittlung der Objektivverzerrung
(Abb. aus: Camera Calibration. `http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm`, letzter Aufruf: 07.06.2010)

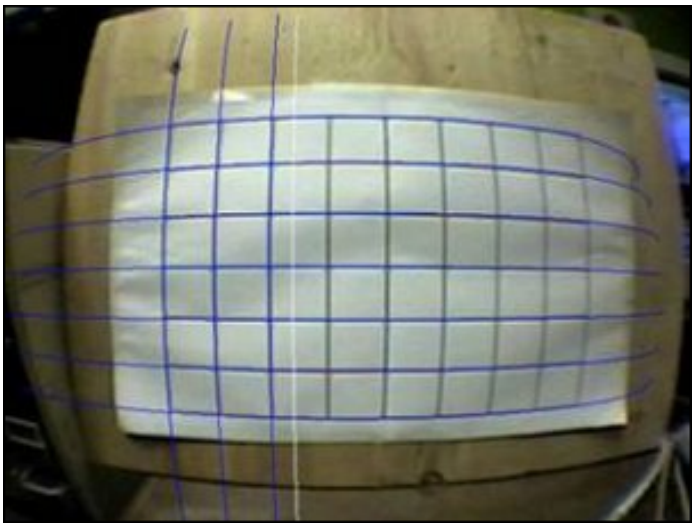


Abbildung 7.: Ermittlung der Brennweite
(Abb. aus: Camera Calibration. `http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm`, letzter Aufruf: 07.06.2010)

Für die Erkennung der Marker wird zunächst das Eingangsbild in ein Graustufenbild umgewandelt. Das ist notwendig, um anschließend eine Schwellwertprüfung für jedes Pixel durchzuführen. Liegt der Pixelwert unter dem Schwellwert, wird der Wert des Pixels auf schwarz gesetzt. Analog dazu wird der Pixelwert auf weiß gesetzt, wenn er über dem Schwellwert liegt. Anschließend werden zusammenhängende schwarze Pixel gruppiert. Die daraus entstehenden Objekte werden einer Konturdetektion unterzogen. Aus diesen Konturen wird eine durchgängige Kante angenähert. Sind die Kanten des Objektes vierseitig, wird dieses Objekt als Marker erkannt. Die Koordinaten der vier Ecken werden ermittelt. Daraus kann das System nun die Position der Kamera in Bezug auf den Marker berechnen. Anhand dieser Daten können somit die virtuellen Elemente platzieren.[36,37,38,39]

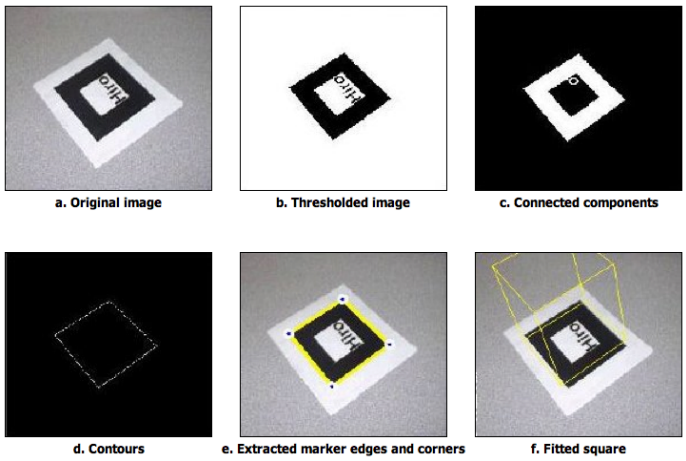


Abbildung 8.: Prinzip des ARToolKit Algorithmus
(Abb. aus: Computer-Vision-Algorithm. `http://www.hitl.washington.edu/artoolkit/documentation/vision.htm`,letzter Aufruf: 12.05.2010)

Im nächsten Schritt wird der Inhalt des Markers analysiert. Damit das System einen Marker identifizieren kann, muss vorher eine sogenannte Pattern-Datei angelegt werden. Diese enthält den Teil eines Markers, der innerhalb der schwarzen Umrandung liegt. Dieses innere Muster ist in der Pattern-Datei in zwölf verschiedenen Ausrichtungen gespeichert. Damit nun die Muster mit dem erkannten Marker im Videobild verglichen werden können, muss dieser erst normalisiert werden, das heißt er wird in

[36] vgl. CAWOOD/FIALA 2008, 286 ff.
[37] vgl. KATO/BILLINGHURST 1999, 4 f.
[38] vgl. How does ARToolKit work? Aufruf: 12.05.2010
[39] vgl. Computer-Vision-Algorithm Aufruf: 12.05.2010

seiner Perspektive und Größe korrigiert. So ist es möglich, verschiedene Marker in einer Szene anhand ihrer Muster zu identifizieren.[40,41] Die Erstellung einer solchen Pattern-Datei geht also jeder *ARToolKit* Anwendung voraus. Um aus einer Bilddatei eine Pattern-Datei zu erstellen, gibt es mehrere Möglichkeiten. Das *ARToolKit* beinhaltet ein Kommandozeilentool zur Erstellung von Pattern-Dateien. Eine weitaus komfortablere Lösung bietet der *ARToolKit Marker Generator Online*.[42] Mit diesem Programm können Bilddateien von selbst erstellten Markern geladen und die dazu gehörige Pattern-Dateien erstellt werden. Es ist aber auch möglich, den ausgedruckten Marker direkt in die Kamera zu zeigen. Das Programm erkennt diesen und erstellt die Pattern-Datei. Letztere Methode hat den Vorteil, dass die Lichtverhältnisse und die Eigenschaften der Kamera mit in die Pattern-Datei einfließen. Die Erstellung von eigenen Markern und den dazugehörigen Pattern-Dateien ist unter 3.2.1.1 beschrieben.

Die Vorteile von *ARToolKit* liegen im einfachen Tracking und in der leichten Registrierung der virtuellen Objekte anhand der Marker. Sofern der Marker korrekt erkannt wird, werden die virtuellen Objekte genau positioniert. Weiter können die Marker mit einfachen Mitteln wie einem Drucker erstellt werden. Auch die Anforderungen an Hardware wie die Kamera und an die Rechenleistung des Computers sind im Vergleich zu markerlosen Registrierungsverfahren gering.

Die Nachteile liegen in der geringen Robustheit des Systems. So werden Marker bei schlechten Lichtverhältnissen nicht zuverlässig erkannt, da hier das erläuterte Schwellwertverfahren keine korrekten Ergebnisse liefert. Und es werden teilweise verdeckte Marker nicht erkannt, da die Kanten nicht durchgängig und zusammenhängend sind.

Bei anderen Markersystemen zum Beispiel *ARTag* sind diese Probleme nicht mehr vorhanden. Durch andere Bildverarbeitungsprozesse ist ein Erkennen von teilweise verdeckten Markern möglich, schlechte Lichtverhältnisse spielen eine geringere Rolle. Während bei *ARToolKit* die Quadrate mittels einer Schwellwertoperation gesucht werden, benutzt *ARTag* eine Kanten- und Liniendetektion. Mit dieser ist es möglich, auch unterbrochenen Kanten durch Annäherung und Schätzung zu finden. *ARTag* ist also robuster, hat aber dadurch eine schlechtere Performance als *ARToolKit*, da die Bildverarbeitungsschritte aufwendiger zu berechnen sind.[43]

[40] vgl. CAWOOD/FIALA 2008, 286 ff.
[41] vgl. KATO/BILLINGHURST 1999, 4 f.
[42] siehe `http://flash.tarotaro.org/blog/2009/07/12/mgo2/`
[43] vgl. CAWOOD/FIALA 2008, 279 ff.

Die Zuverlässigkeit der Erkennung ist auch von der Größe des physikalischen Markers abhängig. Je größer der gedruckte Marker, desto größer ist die Entfernung, in der das System den Marker noch erkennen kann. Eine Gegenüberstellung zwischen Größe und möglicher Entfernung des Markers von der Kamera gibt Tabelle 2.1. Auch das innere Muster eines Markers ist ausschlaggebend für die Robustheit seiner Erkennung. Zum einen darf das Muster nicht symmetrisch sein, da sonst die Ausrichtung nicht eindeutig bestimmt werden kann. Zum anderen darf das gewählte Muster nicht zu komplex sein, da sonst bei zunehmender Entfernung des Markers zur Kamera das Muster nicht mehr erkannt werden kann. Am besten eignen sich Muster mit großen schwarzen und weißen Bereichen.[44,45,46] Die schlechte Erkennung von *ARToolKit* Markern bei Verdeckung lässt sich aber auch positiv nutzen. So könnte man aus dieser Eigenschaft eine Art Interface entwickeln. Wie bei einem Button kann ein Ereignis ausgelöst werden, in dem der Benutzer einen Marker mit seiner Hand verdeckt.

Größe des Musters in *cm*	Mögliche Entfernung zur Kamera in *m*
7,0	0,4
8,9	0,63
7,8	0,86
18,7	1,27

Tabelle 2.1.: Verhältnis Mustergröße zu möglicher Kameraentfernung
(Tabelle aus: How does ARToolKit work? `http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm`, letzter Aufruf: 14.05.2010)

2.3.2. Markerlose Registrierung

Die markerlose Registrierung nutzt keinerlei in die reale Welt zusätzlich eingebrachte Markierungen. Die Informationen zur korrekten Registrierung der virtuellen Objekte müssen allein aus den Bilddaten gewonnen werden. Daraus resultieren einige Probleme. Zum einen sind die Anforderungen an die Leistung eines Systems wesentlich höher als bei der markerbasierten Registrierung. Zum anderen hängt die Genauigkeit der Registrierung von der Genauigkeit der Bildanalyseverfahren und des Tracking

[44] vgl. CAWOOD/FIALA 2008, 279 ff.
[45] vgl. How does ARToolKit work? Aufruf: 12.05.2010
[46] vgl. Computer-Vision-Algorithm Aufruf: 12.05.2010

ab. Hier muss meist ein Kompromiss zwischen Performance und Genauigkeit eingegangen werden, um die Echtzeitfähigkeit des Systems gewährleisten zu können. Die markerlose Registrierung wird in zukünftigen Augmented Reality Anwendungen eine immer größere Rolle spielen. Der Blick des Benutzers wird nicht durch künstlich eingebrachte Elemente gestört. Dadurch ist das Gefühl, dass reale und virtuelle Objekte zu einer neuen Realität verschmelzen, weitaus größer. Bei den Verfahren der markerlosen Registrierung kann man zwischen monokularen und binokularen Techniken unterscheiden. Je nach dem, ob ein Stereobild oder ein Monobild der Szene zur Analyse verwendet wird. Die monokularen Verfahren bilden die Grundlage für die bildanalytischen Verfahren und sollen deshalb weiter erläutert werden.

2.3.2.1. Bildanalytische Verfahren

Die markerlose Registrierung baut hauptsächlich auf der Analyse der Bilddaten auf. Andere Möglichkeiten der markerlosen Registrierung wie zum Beispiel durch Ortung (siehe 2.3.2.2) dienen je nach Art der Anwendung als Ergänzung eines solchen Systems. Wenn Augmented Reality Anwendungen eine markerlose Registrierung verwenden, spricht die Literatur auch von markerloser Augmented Reality. Bei markerloser Augmented Reality werden keine zusätzliche Markerobjekte in die Szene gebracht. Vielmehr wird jeder Teil der realen Umgebung als potentieller Marker gesehen. Die Aufgabe des Systems ist es also markante Punkte im Bild zu erkennen, diese zu verfolgen und daraus Informationen über die Kameraposition und Orientierung zu erhalten. Teichrieb et al. unterscheiden dabei zwischen den Techniken Model Based und Structure from Motion.[47] (Abb. 9)

Bei der Model Based bzw. modellbasierten markerlosen Augmented Reality werden 3D-Modelle von realen Objekten für die Registrierung und das Tracking verwendet. Diese Modelle müssen zuvor manuell oder automatisiert erstellt werden. Anschließend werden sie mit ihren realen Konterparts in Lage und Orientierung in Übereinstimmung gebracht. Diese Initialisierung geschieht manuell durch den Nutzer. Dadurch wird die Lage und Orientierung des realen Objektes bekannt und der Tracker weiß, welches Objekt verfolgt werden soll. Die Vorteile eines solchen Verfahrens liegen in dem hohen Wissen, dass das System über die modellierten realen Objekte besitzt. Durch das 3D-Modell ist dem System nicht nur die Position und Orientierung

[47] TEICHRIEB/DO MONTE LIMA/APOLINÁRIO ET AL. 2007, 1

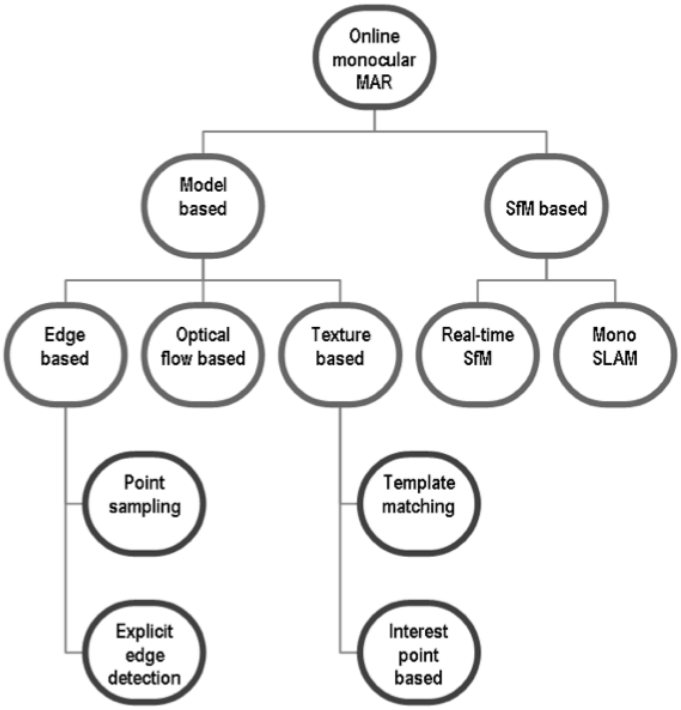


Abbildung 9.: Monokulare markerlose Augmented Reality
(Abb. aus: TEICHRIEB/DO MONTE LIMA/APOLINÁRIO ET AL. 2007, 2)

bekannt, sondern auch die Form des realen Objektes. Somit sind Interaktionen zwischen virtuellen und realen Objekten möglich und es können Verdeckungsproblematiken (siehe 2.4) gelöst werden. Die Nachteile liegen in der manuellen Initialisierung des Trackers. Wenn das Tracking fehlschlägt, weil etwa das zu verfolgende Objekt im Bild nicht mehr sichtbar ist, muss der Tracker manuell neu initialisiert werden.

Modellbasierte Verfahren können in drei Gruppen eingeteilt werden. Verfahren basierend auf Kantendetektion, Texturinformationen und Verfahren, die den optischen Fluss (optical flow) einer Bildsequenz analysieren.[48]

Structure from Motion ist angelehnt an den *Kinetic Depth Effect* (siehe 2.4.1). Es wird versucht, anhand der Bewegung von Objekten und der daraus entstehenden perspektivischen Verzerrungen eine dreidimensionale Rekonstruktion abzuleiten. Das Verfahren besteht dabei aus drei grundlegenden Schritten. Im ersten Schritt wird im Bild nach Merkmalen gesucht. Diese werden mittels eines Trackingverfahrens verfolgt. Anhand der gewonnenen Daten werden anschließend die Kameraparameter berechnet. Für eine korrekte Registrierung ist also kein a priori Wissen über die Szene notwendig. Alle relevanten Daten werden aus der Bildsequenz gewonnen. Weiter ist die Lösung von Verdeckungsproblemen (siehe 2.4) möglich. Ein großer Nachteil ist der große Berechnungsaufwand der verschiedenen Schritte. Auch hier müssen in Hinblick auf die Echtzeitfähigkeit des Systems Kompromisse in den Punkten Genauigkeit und Robustheit eingegangen werden.[49]

2.3.2.2. Geographische Lokation

Eine weitere Möglichkeit der markerlosen Registrierung oder zumindest diese zu unterstützen bieten geographische Daten. Diese beinhalten zum Beispiel Informationen über Position, Höhenlage, Blick- und Bewegungsrichtung oder Geschwindigkeit. Diese Daten werden mit einem GPS-Empfänger ermittelt. Aber auch eine Ortung über das GSM-Netz ist möglich. Je nach Anwendung ist die Genauigkeit der Positionsbestimmung ein Problem.[50] Vor allem bei mobilen Augmented Reality Anwendungen auf PDAs oder Mobiltelefonen bietet sich diese Technik an. Ein Beispiel

[48] vgl. TEICHRIEB/DO MONTE LIMA/APOLINÁRIO ET AL. 2007, 2 ff.

[49] vgl. TEICHRIEB/DO MONTE LIMA/APOLINÁRIO ET AL. 2007, 5 ff.

[50] Bei handelsüblichen GPS Empfängern und Mobiltelefonen liegt die Genauigkeit zwischen 2 und 13 Metern. (vgl. GPS-Global Positioning System. `http://www.elektronik-kompendium.de/sites/kom/1201071.htm`, letzter Aufruf: 15.06.2010)

hierfür ist ein virtueller Stadtführer. Man richtet die Kamera des Mobiltelefons auf ein Gebäude oder Bauwerk. Die Software erkennt dieses und erweitert das Videobild. Es können zum Beispiel 3D-Modelle von historischen Gebäuden in Echtzeit angezeigt werden, wie bei der Anwendung *Archeoguide*[51]. Nun wäre ein rein auf Bildanalyse aufgebautes Verfahren hier schwer umsetzbar, da es auf der Welt unzählige Häuser, Bauwerke oder Plätze gibt, die von Interesse sein könnten. Durch Daten wie geographische Position und Blickrichtung ist es jedoch möglich, die Anzahl der möglichen Objekte auf einige wenige einzugrenzen. Diese können dann anhand einer Bilddatenbank verglichen und identifiziert werden.

2.4. Darstellung von virtuellen Objekten in einer realen Umgebung

Das langfristige Ziel von Augmented Reality ist die nahtlose Verschmelzung von virtuellen Elementen mit der realen Welt. Der Nutzer soll nicht mehr zwischen beiden unterscheiden können, sondern die virtuellen Elemente als scheinbar real wahrnehmen. Es soll also eine räumliche Präsenz der Augmented Reality Anwendung beim Nutzer erzeugt werden.[52] Um dieses Ziel zu erreichen, müssen verschiedene wahrnehmungsspezifische Aspekte berücksichtigt werden. Nicht jede Augmented Reality Anwendung muss zwingend das Kriterium der nahtlosen Verschmelzung erfüllen. Bei manchen Anwendungen steht eher die Präzision und Qualität der Informationen im Vordergrund als die Tatsache, eine perfekte Illusion zu schaffen. Beispiel hierfür sind bereits erwähnten medizinischen Anwendungen oder auch solche in der Industrie. Dennoch steigert eine authentische Darstellung auch die Akzeptanz des Benutzers gegenüber Augmented Reality.[53]

2.4.1. Darstellungsfaktoren

In Augmented Reality Anwendungen steht die reale Welt im Vordergrund. (siehe 2) Der virtuelle Anteil muss sich in seiner Darstellung an die Gegebenheiten und Gesetzmäßigkeiten dieser anpassen. Zum einen sind das visuelle Gesetzmäßigkeiten. aber auch Gesetzmäßigkeiten anderer Sinne können davon betroffen sein. Zum Bei-

[51] siehe `http://archeoguide.intranet.gr/project.htm`
[52] vgl. ABAWI 2005 b, 15
[53] vgl. ABAWI 2005 b, 15 ff.

spiel ist der Klang eines Instruments desto leiser, je größer seine Entfernung zum Zuhörer ist. Gegenstand dieser Arbeit soll jedoch ausschließlich die visuelle Augmented Reality sein.

Die menschliche visuelle Wahrnehmung ist sehr empfindlich gegenüber widersprüchlichen Informationen. Wenn ein Objekt den Gesetzmäßigkeiten wie Verdeckung, Schatten oder Reflexionen nicht folgt, wird dieses vom Benutzer sofort als nicht real erkannt. Diese Faktoren dienen hauptsächlich der Tiefenwahrnehmung von Objekten. Drascic und Milgram unterscheinden dabei zwischen Darstellungs- und Bewegungsfaktoren sowie zwischen physiologischen und binokularen Faktoren.[54]

Darstellungsfaktoren sind alle Faktoren, die das Auge direkt aus dem wahrgenommenen Bild entnimmt. Dazu zählen die Größe und die perspektivische Darstellung des Objektes und der Texturen aber auch die Helligkeit und der Schattenwurf von Objekten. Weiter nimmt mit zunehmender Entfernung nicht nur die Größe sondern auch die erkennbaren Details ab. Auch die Farbe und Helligkeit ist abhängig von der realen Welt. Bei farbiger Beleuchtung muss sich das auch auf das virtuelle Objekt auswirken. Zudem ist die eigene Schattenbildung und der Schattenwurf durch und auf andere virtuelle und reale Objekte von Bedeutung. All das zeigt, wie schwierig eine realistische Koexistenz von virtuellen und realen Objekten ist. Zwar sind Faktoren wie Größe und Perspektive meist durch eine korrekte Registrierung bereits abgedeckt, jedoch speziell die Problematik von Schatten und Verdeckung erfordern eine genauere Modellierung der realen Welt. Diese werden wegen ihrer großen Bedeutung für Augmented Reality Anwendungen unter 2.4.2. näher betrachtet.[55]

Aus der Bewegung eines Objektes lassen sich Informationen über seine Tiefe im Raum gewinnen, dass nennt man Bewegungsfaktoren. So bewegen sich Objekte, die weiter von uns entfernt sind, scheinbar langsamer als Objekte, die direkt vor uns sind. Wenn sich der Benutzer bewegt, also sich der Blickwinkel ändert, bewegen sich nahe Objekte in die entgegengesetzte Richtung. Objekte in der Ferne scheinen uns zu folgen. Auch diese Effekte der Tiefenwahrnehmung werden bereits durch eine korrekte Registrierung berücksichtigt. Ein weiterer interessanter Effekt ist der *Kinetic Depth Effect*. Dabei wird das Bild in seiner Perspektive immer parallel zum Blickwinkel des

[54] vgl. DRASCIC/MILGRAM 1996, 126 ff.
[55] vgl. DRASCIC/MILGRAM 1996, 126

Betrachters geändert. Solang der Betrachter in Bewegung ist, sich also auch das Bild immer leicht bewegt, scheint dieses eine räumliche Tiefe zu besitzen.[56,57]

Physiologische Faktoren betreffen die Funktion des Auges. Das menschliche Auge kann immer nur eine Tiefenebene scharf abbilden. Wie bei einer Kamera kann es seinen Fokus verändern. Die dadurch entstehenden unscharfen Bereiche geben weitere Tiefeninformationen. Vor allem bei See Through Displays ist der veränderliche Fokus des Auges ein Problem. (siehe 2.2)

Binokulare Faktoren meinen vor allem Effekte, die aus dem stereoskopischen Sehen hervorgehen. Durch den Abstand unserer Augen entstehen zwei leicht versetzte Bilder. Aus diesem Versatz lassen sich wichtige Informationen über die räumliche Tiefe gewinnen, ohne den Blickwinkel zu ändern. Dabei gibt es einen Zusammenhang zwischen Objekttiefe und Bildversatz, wobei dieser aber auch vom jeweiligen Fokus abhängig ist.

2.4.2. Verdeckung und Schatten

Ein weiteres Problem von Augmented Reality Anwendungen ist die Interposition von Objekten. Liegt ein Objekt räumlich vor einem anderen, wird dieses ganz oder teilweise verdeckt. Um dies korrekt darstellen zu können, benötigt man also Informationen über den Abstand der Objekte vom Betrachter bzw. der Objekte untereinander. Das Objekt mit der größeren Entfernung wird von dem mit der geringeren Entfernung verdeckt, sobald diese sich überschneiden. Handelt es dabei um zwei virtuelle Objekte, kann man die korrekte Darstellung mathematisch berechnen.[58] Die Verdeckung von realen durch virtuelle Elemente ist technisch gesehen kein Problem, da nur das virtuelle Bild über das reale Bild gelegt werden muss. Bei der Verdeckung von virtuellen durch reale Objekte darf hingegen nur der Teil des virtuellen Objekts dargestellt werden, der nicht von der Verdeckung betroffen ist. Dies setzt ein Wissen über die Form, Größe und dreidimensionale Position der realen Objekte voraus.

[56] vgl. DRASCIC/MILGRAM 1996, 126

[57] siehe Beispiel: `http://www.youtube.com/watch?v=8SDGG9HhbgQ&feature=related`

[58] siehe zum Beispiel Z-Buffer: `http://de.wikipedia.org/wiki/Z-Buffer`

Auch das Material des Objekts könnte von Bedeutung sein, wenn es zum Beispiel einen gewissen Grad an Transparenz aufweist. [59,60]

Für die korrekte Schattendarstellung müssen die betroffenen realen Objekte in Form, Größe und Position dem System bekannt sein und die Eigenschaften der realen Lichtquellen aus den Bilddaten entnommen werden. Zu diesen Eigenschaften zählen zum Beispiel Position, Farbe, Intensität und Streuung des Lichtes. Die korrekte Berechnung von Schatten- und Schattierungseffekten ist in dem Umfeld der Augmented Reality sehr rechenaufwendig. Um eine reale Darstellung von Schatten zu gewährleisten, müssten theoretisch alle realen Objekte in einer Szene, die nicht zum Hintergrund gehören, vollständig modelliert sein.

2.4.2.1. Verfahren für die korrekte Verdeckungsdarstellung

Um das Verdeckungsproblem zu lösen, gibt es verschiedene Ansätze. Daniel Abawi hat in seiner Arbeit verschiedene Verfahren für die korrekte Darstellung von Verdeckung in Augmented Reality aufgeführt.[61] Hierbei unterscheidet Abawi zwischen manuellen, halbautomatischen und automatischen Verfahren. Eine genauere Beschreibung und Bewertung der Verfahren ist in der Arbeit von Daniel Abawi zu finden. Die im Folgenden genannten Verfahren sind nicht als allgemeingültige fertige Lösungen für Verdeckungsdarstellungsproblematik in Augmented Reality Anwendungen zu verstehen. Vielmehr bieten sie grundlegende Ansätze, anhand denen man für jeweilige Anwendungen spezifische Verfahren entwickeln kann.

Die einfachste Lösung, aber auch gleichzeitig die am wenigsten flexibelste, ist die manuelle Erstellung von zweidimensionalen Masken. Hierbei wird eine Binärmaske erstellt, die nur die Zustände *verdecken* und *nicht verdecken* kennzeichnet. Überschneidet sich nun ein virtuelles Element mit einem *verdecken* Bereich, werden die betroffenen Teile nicht dargestellt.[62]

Ein halbautomatisches Verfahren ist die dynamische Verdeckung vor statischen Hintergründen. Hierbei ist es möglich, ein virtuelles Element zwischen dynamische, rea-

[59] vgl. ABAWI 2005 b, 19 f.
[60] vgl. DRASCIC/MILGRAM 1996, 130 ff.
[61] vgl. ABAWI 2005 b
[62] vgl. ABAWI 2005 b, 28 f. nach NAKAMAE/HARADA/ISHIKAZO ET AL. 1986

le Objekte und einen bekannten Hintergrund zu platzieren. Dabei werden zuvor dem System alle vorkommenden Hintergrundbilder mitgeteilt. In der Anwendung wird dann durch Vergleich das aktuell gezeigte Hintergrundbild identifiziert. Das zuvor gespeicherte Bild wird perspektivisch an das Kamerabild angeglichen. Nun kann das System das aktuelle Kamerabild durch einfach Subtraktion der Pixelwerte mit dem gespeicherten vergleichen. An Stellen, an denen die Bilder gleiche Werte haben, ergibt die Subtraktion Null. An den Stellen, wo ein reales Objekt den Hintergrund überlagert, ergibt die Subtraktion einen Wert verschieden von Null. An diesen Stellen findet dann eine Verdeckung statt.[63]

Ein weiteres halbautomatisches Verfahren arbeitet mit vorkonstruierten 3D-Modellen, die vorher von realen Objekten einer Augmented Reality Szene erstellt werden. Diese Modelle werden dann vom Benutzer an der Stelle in der Szene platziert, an der sich das korrespondierende reale Objekt befindet und in Größe und Ausrichtung an das reale Objekt angepasst. Dadurch werden die Informationen über Position, Ausrichtung und Tiefe eines realen Objektes auf das 3D-Modell übertragen. Somit ist eine korrekte Verdeckungsdarstellung gegenüber anderen virtuellen Elementen möglich.[64]

Bei automatisierten Verfahren steht die Erstellung einer Tiefenkarte bzw. Depth Map für ein Bild bzw. für bestimmte reale Objekte imVordergrund. Eine Tiefenkarte, bildlich dargestellt, entspricht einem Graustufenbild. Jeder Pixelwert repräsentiert dabei eine bestimmte Entfernung von der Kamera und somit eine bestimmte Tiefe im Raum. Dabei geben Bilder aus unterschiedlichen Perspektiven von einer Szene Aufschluss über die Tiefe der enthaltenen Objekte. Dies allein würde jedoch für eine korrekte Verdeckungsdarstellung nicht ausreichen, da die Tiefeninformationen den Objekten zugeordnet werden müssen. Andernfalls könnte eine Durchdringung von realen durch virtuelle Objekte die Folge sein, was zu einer unrealistischen Darstellung führen würde.[65]

[63] vgl. ABAWI 2005 b, 29 nach FISCHER/REGENBRECHT/BARATOFF 2003
[64] vgl. ABAWI 2005 b, 29 nach BREEN/WHITAKER/ROSE ET AL. 1995
[65] vgl. ABAWI 2005 b, 29 ff.

2.5. Beispielanwendungen

Der Fokus richtet sich bei den Beispielen auf Anwendungen, die bereits für den Endverbraucher erhältlich sind oder sich schon in der Anwendung befinden. Um einen Eindruck zu bekommen, wie vielfältig Augmented Reality sein kann, sollen Beispiele aus den Bereichen Medizin, Werbung und mobiler Augmented Reality Anwendungen gezeigt werden. Bei einigen der genannten Beispiele sind die Merkmale von Augmented Reality laut Definition (siehe 2) nur teilweise gegeben. Gerade die Dreidimensionalität und Echtzeitfähigkeit sind nicht immer vorhanden. Dennoch sind die Beispiele nach eigener Einschätzung als Augmented Reality anzusehen, da sie die grundlegenden Züge von Augmented Reality enthalten. Beispielanwendungen für Augmented Reality in RIAs sind im Kapitel 3 unter dem Abschnitt 3.6 aufgeführt. Weitere Beispiele für die Anwendung von Augmented Reality aus anderen Bereichen finden sich bei Friedrich (Hrsg.) 2004, Abawi 2005 a und Abawi 2005 b.

2.5.1. Lego Digital Box

Die *Lego Digital Box* ist eine markerbasierte Augmented Reality Anwendung zur dreidimensionalen Präsentation von *Lego* Produkten. Dabei kann der Kunde im Laden die Verpackung eines *Lego* Spielzeugs in die Kamera der *Digital Box* halten. Das System erkennt den aufgedruckten Marker und überlagert in Echtzeit ein 3D-Modell des Spielzeugs. (Abb. 10) Die Produktpräsentation wird also durch eine dreidimensionale Darstellung erweitert. Zusätzlich zu den auf der Verpackung abgedruckten Bildern des Produkts, bekommt der Kunde einen räumlichen Eindruck, wie das zusammen gebaute Spielzeug am Ende aussieht.[66]

[66] vgl. The Lego Group to Boost Retail with Metaio. Pressemittelung der metaio GmbH, Dezember 2008, `http://www.metaio.com/fileadmin/homepage/Presse/Dokumente/Pressemitteilungen/English/E_2009_01_15_PR_LEGO.pdf`, letzter Aufruf: 09.06.2010

Abbildung 10.: Lego Digital Box Terminal
(Abb. aus: The Lego Group to Boost Retail with Metaio. Pressemitteilung der metaio GmbH, Dezember 2008, Seite 2, `http://www.metaio.com/fileadmin/homepage/Presse/Dokumente/Pressemitteilungen/English/E_2009_01_15_PR_LEGO.pdf`, letzter Aufruf: 09.06.2010)

2.5.2. Augmented Reality bei der minimal invasiven Laparoskopie

Ein Beispiel für eine Projection Based Augmented Reality Anwendung ist die Methode des Chirurgen Dr. Maki Sugimoto eine minimal invasive Laparoskopie[67] mit Augmented Reality Hilfestellung durchzuführen. Dabei werden zuvor aufgenommene Bilder vom Inneren des Körpers mit dem Beamer auf den Bauch des Patienten projiziert. (Abb. 11) Die Tiefe der Darstellung lässt sich dabei stufenlos mit einer Fernbedienung regeln. Die Registrierung wird anhand des Bauchnabels vorgenommen. Die Position des Nabels auf dem projizierten Bild wird manuell mit dem des Patienten in Übereinstimmung gebracht. Auf die Verwendung von Kameras im Bauchinneren kann auch durch Augmented Reality nicht verzichtet werden. Dennoch bietet Augmented Reality in diesem Fall eine bessere, vorherige Lokalisation des Krankheitsherdes.[68,69]

2.5.3. Augmented Driving und Wikitude Drive

Augmented Driving ist eine Augmented Reality Anwendung der Firma *imaGinyze* für das *iPhone*. *Augmented Driving* zeigt dabei dem Fahrer zusätzliche Informationen auf dem Live Videobild[70] des Smartphones an. (Abb. 12) Eines der wesentlichen Funktionen ist die Erkennung von anderen Fahrzeugen. So wird die Entfernung zu dem voraus fahrenden Fahrzeugen gemessen und ggf. ein Warnsignal ausgelöst, wenn der Abstand einen bestimmten Grenzwert unterschreitet. Außerdem wird auch der

[67] Minimal invasive Operation bezeichnet einen operativen Eingriff mit kleinstmöglichen Verletzung an Haut und Weichteilen. Dies wird durch möglichst kleine Einschnitte erreicht. Bei der Laparoskopie orientiert sich dabei der Arzt anhand von Videobildern, die mit Kameras vom inneren des Körpers während der Operation erstellt werden. (vgl. JENSEN `http://www.frauenklinik-uni-bochum.de/pdf/lsk_zentrum.pdf`, letzter Aufruf: 08.06.2010)

[68] vgl. Creating Roadmaps for Surgeons. Using the Mac for Better Surgical Navigation. `http://www.apple.com/science/profiles/maki/`, letzter Aufruf: 09.06.2010

[69] vgl. Minimalinvasive Laparoskopie mit OsiriX. `http://aycandigital.blogspot.com/2009/02/minimalinvasive-laparoskopie-mit-osirix.html`, letzter Aufruf: 09.06.2010

[70] Wegen der begrenzten Rechenleistung ist die Anwendung aktuell auf eine Bildwiederholrate von 6-11 Bilder pro Sekunde begrenzt. Somit ist es formal gesehen keine Echtzeitanwendung.

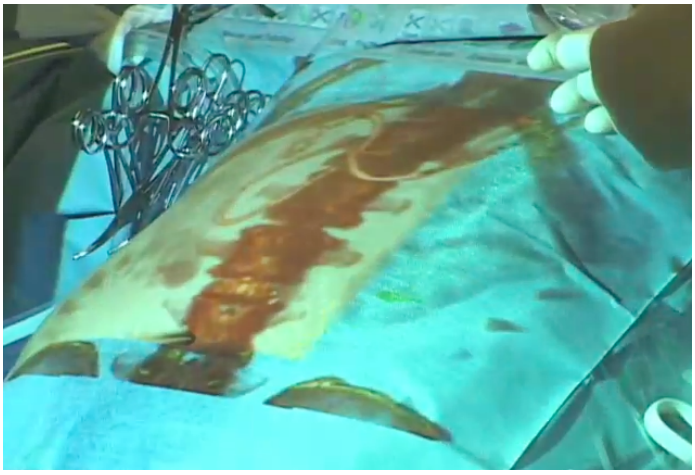


Abbildung 11.: Minimal invasive Laparoskopie unter Zuhilfenahme von Augmented Reality
(Abb. aus: Creating Roadmaps for Surgeons. Using the Mac for Better Surgical Navigation. `http://www.apple.com/science/profiles/maki/`, letzter Aufruf: 09.06.2010)

Wechsel der Fahrspur vom System erfasst. Dies kann nützlich sein, wenn man durch Ablenkung ungewollt von der Fahrspur abkommt.[71]

Eine weitere mobile Augmented Reality Anwendung ist *Wikitude Drive*. Diese Anwendung für *Android* Smartphones ist eine Navigationssoftware. Im Gegensatz zu herkömmlichen Navigationsgeräten wird jedoch nicht eine abstrakte Karte angezeigt, sondern die Routeninformationen direkt über das Live Videobild des Smartphones gelegt. (Abb. 13) So hat der Fahrer immer die Straße im Blickfeld und die Orientierung fällt leichter.[72]

[71] vgl. Augmented Driving. `http://www.imaginyze.com/Site/Welcome.html`, letzter Aufruf: 08.06.2010

[72] vgl. Wikitude Drive: Never Take your Eyes of the Road again. `http://www.wikitude.org/drive-2`, letzter Aufruf: 08.06.2010

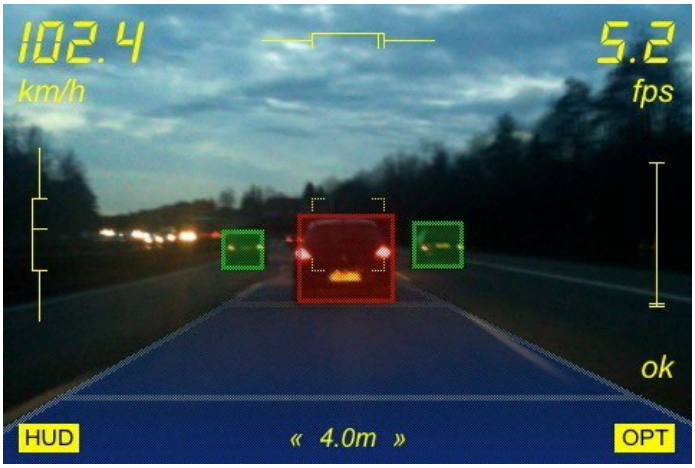


Abbildung 12.: Screenshot aus der Anwendung *Augmented Driving*
(Abb. aus: Augmented Driving Screenshots. `http://www.imaginyze.com/Site/Screenshots.html#4`, letzter Aufruf: 08.06.2010)

Abbildung 13.: Screenshot aus der Anwendung *Wikitude Drive*
(Abb. aus: Wikitude Drive: Never Take your Eyes of the Road again. `http://www.wikitude.org/drive-2`, letzter Aufruf: 08.06.2010)

3. Umsetzung von Augmented Reality in Rich Internet Applications

Die Anwendungsmöglichkeiten von Augmented Reality sind sehr vielfältig. Die visuelle Augmented Reality steht dabei im Vordergrund vieler Anwendungen. Das Internet bietet als vorrangig visuelles Medium ein breites Spektrum an Anwendungsmöglichkeiten. Vor allem Onlineshops können von dieser Technologie profitieren. Augmented Reality bietet eine völlig neue Form der Produktpräsentation. Während man bisher nur erahnen konnte, ob ein Kleidungsstück optisch zum Träger passt, kann mit Augmented Reality eine virtuelle Anprobe vorgenommen werden. Eine weitere Möglichkeit ist das virtuelle Platzieren von Möbelstücken im Raum. (siehe 3.6.1) Auch in der Werbebranche kann Augmented Reality genutzt werden. So können Printanzeigen von Produkten durch einen abgedruckten Marker ergänzt werden. Der Kunde kann sich dann mittels Computer oder Smartphone zusätzliche Informationen wie zum Beispiel ein 3D-Modell des Produkts anzeigen lassen. (siehe 3.6.3) Einige Beispiele für bereits entwickelte Augmented Reality RIAs sind unter 3.6 aufgeführt.

Die Umsetzung von Augmented Reality als reine Webanwendung ist zum heutigen Zeitpunkt nicht möglich, da die grundlegenden Voraussetzungen für Augmented Reality (siehe 2) nicht erfüllt werden können. Als serverseitige Programme sind Webanwendungen nicht in der Lage, in Echtzeit zu interagieren. Um dennoch Augmented Reality im Internet nutzen zu können, bedient man sich so genannter Rich Internet Applications. Dies sind Programme, die Internettechnologien nutzen, jedoch clientseitig ausgeführt werden. Dadurch steht mehr Rechenleistung zur Verfügung, wodurch wesentlich komplexere Anwendungen realisierbar sind. Weiter müssen nur für bestimmte Funktionalitäten Daten vom Webserver angefordert und übermittelt werden. Plattformen für die Entwicklung von RIAs sind zum Beispiel *Adobe Flash*, *Microsoft Silverlight* oder *Java*. Theoretisch können auf diesen Plattformen auch mobile Augmented Reality Anwendungen erstellt werden. Jedoch wäre eine Entwicklung als native Anwendung für das jeweilige Betriebssystem wesentlich performanter, was bei der begrenzten Rechenleistung von mobilen Geräten von großer Bedeutung sein kann. Aus diesem Grund beschränkt sich dieses Kapitel auf Desktopanwendungen. Es soll jedoch darauf hingewiesen werden, dass RIAs prinzipiell auch auf mobilen

Endgeräten funktionieren. In diesem Kapitel wird von einer Anordnung ausgegangen, in der eine fest installierte Webcam das Live-Videobild erzeugt.

3.1. Anforderungen an eine RIA-Plattform für visuelle Augmented Reality

Für die Umsetzung von visueller Augmented Reality muss eine RIA-Plattform vier Grundvoraussetzungen erfüllen. Als erstes ist ein Abbild der Realität zu schaffen. Dieses Abbild wird durch ein Live-Videobild repräsentiert, das auf dem Bildschirm des Benutzers dargestellt wird (siehe 2.2.3). Den direkten Zugriff auf die Webcam bzw. eine andersweitig angeschlossene Kamera bieten zum Zeitpunkt dieser Arbeit die Plattformen *Adobe Flash*, *Microsoft Silverlight* und *Java*. Als zweite Voraussetzung müssen die Daten in Echtzeit verarbeitet werden können. Dies ist mit den genannten Systemen möglich. Die dritte Voraussetzung betrifft die dreidimensionale Registrierung. Für eine markerbasierte Registrierung kann zum Beispiel die jeweilige Portierung des *ARToolKit* verwendet werden.[73,74,75] Als vierte Voraussetzung müssen die Plattformen 3D-Objekte darstellen können. Zwar ist bei *Flash* und *Silverlight* keine native Unterstützung der Darstellung von 3D-Modellen implementiert, jedoch lässt sich dies durch die Benutzung von Frameworks wie *Papervision3D* für *Flash* oder *Kit3D* für *Silverlight* realisieren. Hier bietet Java mit der Bibliothek Java3D einen entscheidenden Vorteil. Während bei den beiden anderen Plattformen die 3D-Grafiken durch die CPU berechnet werden, bietet Java3D die Möglichkeit per OpenGL oder Direct3D Berechnungen auf der GPU durchzuführen.

Bei der Umsetzung von Augmented Reality in RIAs ist eine Beschränkung auf Spatial Video See Through Displays für Desktopanwendungen und Hand-Held Video See Through Displays für mobile Anwendungen sinnvoll, da hierfür keine spezielle Hardware erforderlich ist. (siehe 2.2) Anhand dieser Beschränkungen auf bestimmte Displayarten sind der Augmented Reality in RIAs Grenzen gesetzt. So kann eine

[73] ARToolkit für Flash: FLARToolKit `http://www.libspark.org/wiki/saqoosha/FLARToolKit/en`

[74] ARToolkit für Silverlight: SLARToolKit `http://slartoolkit.codeplex.com/`

[75] ARToolkit für Java: NyARToolKit for Java `http://nyatla.jp/nyartoolkit/wiki/index.php?NyARToolkit%20for%20Java.en`

komplette Verschmelzung von realer und virtueller Welt nur schwer statt finden, da der Nutzer immer durch einen Bildschirm auf die erweiterte Szene blickt.

Eine weitere, indirekte Anforderung an eine Plattform ist deren hohe Verbreitung und damit Akzeptanz beim Nutzer. Damit Nutzer Augmented Reality Internetanwendungen akzeptieren, sollten diese nahtlos in eine Webseite integriert werden können. Dies ist mit den genannten möglich. Jedoch muss jeweils ein Browserplugin installiert werden, um die Anwendungen ausführen zu können. Eine zusätzliche Installation im Kontext von Webseiten stellt immer eine Barriere für den Nutzer dar. In der Verbreitung liegt laut Statistiken das *Flash* Plugin (ca. 97%) vor *Silverlight* (ca. 60%) und *Java* (ca. 76%).[76,77] Auch wenn die Werte je nach Quelle variieren und nicht repräsentativ sind, geben sie doch einen Anhaltspunkt über die Verbreitung und somit Akzeptanz einer Plattform beim Benutzer wider.

In den folgenden Abschnitten soll beispielhaft *Adobe Flash* als Plattform für die Erstellung von Augmented Reality in RIAs dienen, da *Flash* die vergleichsweise höchste Verbreitung und damit Akzeptanz beim Benutzer besitzt. Die angeführten Punkte können auch auf die anderen Plattformen übertragen werden, da für alle Plattformen äquivalente Bibliotheken existieren.

3.2. Kalibrierung und Registrierung in Augmented Reality RIAs

Eine Kalibrierung wie beispielsweise bei HMDs ist in Augmented Reality RIAs nicht erforderlich, da keine besonderen Displayanordnungen verwendet werden. (siehe 2.2.1) Die Ermittlung der Kameraparamter (siehe 2.3.1.2) kann je nach Art der verwendeten Registrierungsmethode eine Rolle spielen. Bei auf Bilddaten basierenden Registrierungsverfahren sind die Lichtverhältnisse der Umgebung von Bedeutung und müssen gegebenenfalls angepasst werden. Bei der Registrierung wird wieder zwischen markerbasierter und markerloser Registrierung unterschieden.

[76] Rich Internet Application Statistics. Real World Stats of RIA Plugin Developments. `http://riastats.com/`, letzter Aufruf: 04.06.2010

[77] Rich Internet Applications Market Share. RIA Market Penetration and Global Usage. `http://www.statowl.com/custom_ria_market_penetration.php?1=1&timeframe=ytd&interval=month&chart_id=13&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&timeframe=last_6`, letzter Aufruf; 04.06.2010

3.2.1. Markerbasierte Registrierung in Augmented Reality RIAs

Die bei der Recherche dieser Arbeit gefundenen Augmented Reality RIAs findet die Registrierung häufig anhand sogenannter Pattern-Marker statt. Dabei kommen zum Beispiel die *ARToolKit* Portierung für *Flash* (siehe 3.6) oder die *flare*tracker* Bibliothek (siehe 3.2.1.2) für *Flash* zum Einsatz. Es gibt Marker-Systeme die ohne Pattern-Marker arbeiten wie zum Beispiel *flare*nft*. Bei diesem System kann jedes beliebige Bild als Marker verwendet werden. Das Tracking findet anhand von *Natural Features* statt.(siehe 3.2.1.2) Die vorrangige Verwendung von Markern liegt nach eigener Einschätzung daran, dass damit eine korrekten Registrierung relativ leicht umzusetzen ist. Weiter ist es ohne größeren technischen Aufwand möglich, die benötigten Marker herzustellen. Diese können zum Beispiel als Bestandteil einer Printanzeige in gedruckter Form zum Endverbraucher gelangen. Eine weitere Möglichkeit besteht darin, ein druckfertiges Dokument zum Herunterladen auf einer Webseite anzubieten. Der Nutzer kann sich den Marker dann mit einem handelsübliche Drucker ausdrucken.

3.2.1.1. FLARToolKit, FLARManager und Marker-Generator

FLARToolKit wurde von einem japanischen Programmierer namens *Saqoosha* entwickelt. Dabei handelt es sich um eine Portierung von *NYARToolKit*, dass wiederum eine Portierung des *ARToolKit* für die *Adobe Flash* Plattform ist. Für den nichtkommerziellen Einsatz steht *FLARToolKit* unter der GPL Lizenz und kann somit im Rahmen der Lizenzbestimmungen kostenfrei eingesetzt werden. Die Funktionsweise der Bibliothek ist analog zu der unter 2.3.1.2 beschriebenen Funktionsweise von *ARToolKit*. Es werden vorher definierte Pattern-Marker auf einem Eingangsbild erkannt und verfolgt. Dabei wird die Größe, Position und Orientierung des Markers errechnet und der Marker identifiziert. Die Darstellung von 3D-Modellen ist jedoch nicht in *FLARToolKit* implementiert. Es werden aber die 3D-Engines *Papervision3D*, *Away3D*, *Sandy* und *Alternativa3D* unterstützt.[78]

FLARManager ist ein Framework für *FLARToolKit*, dass von Eric Socolofsky entwickelt wurde. Das Framework erleichtert die Erstellung von Augmented Reality

[78] vgl. What is FLARToolKit. `http://www.libspark.org/wiki/saqoosha/FLARToolKit/en`, letzter Aufruf: 07.06.2010

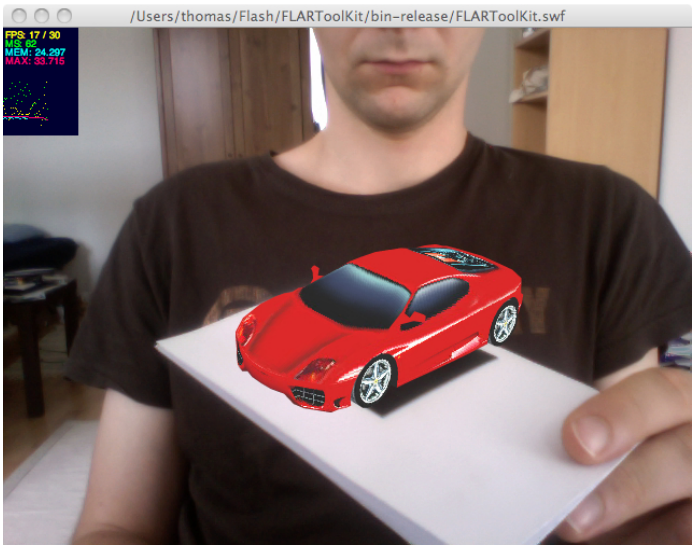


Abbildung 14.: Beispielanwendung mit FLARToolKit

Anwendungen auf Basis von *FLARToolKit*. Die Initialisierung einer *FLARToolKit* Anwendung wird damit auf wenige Zeilen Programmcode reduziert. Die Konfiguration wurde in eine XML Datei ausgelagert. Außerdem sind in dem Frameworks Event-Handler für das Hinzufügen, das Entfernen und für die Änderung der Position eines Markers implementiert, dadurch lässt sich auch mit mehreren Markern komfortabel arbeiten.[79]

In den der Arbeit beiliegenden Beispielen wurde ein individuell erstellter Marker benutzt. (Anlage A) Die zugehörige Pattern-Datei wurde mit dem Programm *ARToolKit Marker Generator Online* des Bloggers *Tarotaro* erstellt.[80] Um eine eigene Pattern-Datei erstellen zu können, benötigt man zunächst eine Bilddatei des Markers. Mit einem Grafikprogramm wird ein schwarzes Quadrat mit einer maximalen Größe von 300x300 Pixeln erstellt. In die Mitte wird ein weißes, nicht symmetrisches Muster gesetzt. Das dient zur späteren Berechnung der Orientierung und der Identifizierung des Markers. Das Muster sollte die in 2.3.1.2 genannten Voraussetzungen erfüllen. Im nächsten Schritt wird das erstellte Bild in den *Marker Generator* geladen. Alternativ kann der ausgedruckte Marker auch per Kamera erfasst werden. Anschließend muss

[79] vgl. Socolofsky, Eric: FLARManager. Augmented Reality in Flash. `http://words.transmote.com/wp/flarmanager/`, letzter Aufruf: 08.06.2010

[80] Marker„s“Generator Online Released! `http://flash.tarotaro.org/blog/2009/07/12/mgo2/`, letzter Aufruf: 08.06.2010

die Auflösung der Pattern-Datei bestimmt werden. Dabei sind Werte von 4x4, 8x8, 16x16, 32x32 und 64x64 möglich. Eine niedrigere Auflösung bietet eine bessere Performance. Dafür sinkt die Erkennungsrate des Markers. Eine höhere Auflösung bietet eine bessere Erkennungsrate, benötigt dafür aber mehr Rechenleistung. Hier sollte je nach Art der Anwendung der beste Kompromiss aus Erkennungsrate und Performance gefunden werden. Im letzten Schritt muss das prozentuale Verhältnis der Größe des inneren Musters zum Marker angegeben werden. (Abb. 15) In der Darstellung nimmt das Muster 50% der Höhe und 50% der Breite des gesamten Markers ein. Abschließend wird die Pattern-Datei erzeugt und gespeichert.[81]

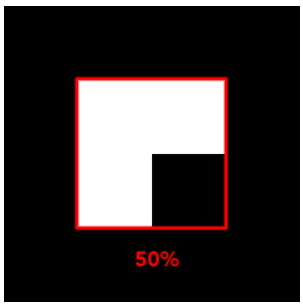


Abbildung 15.: Prozentuales Verhältnis zwischen Mustergröße und Marker (Abb. aus: For Nerds Only: Custom FLAR Markers Explained. `http://www.squidder.com/2009/03/05/for-nerds-only-custom-flar-markers-explained/`, letzter Aufruf: 08.06.2010)

3.2.1.2. flare*tracker und flare*nft

*Flare*tracker* oder auch *Flash Augmented Reality Engine * Tracker* ist eine kommerzielle Software-Bibliothek für *Flash* von der *Imagination Computer Services GesmbH*. Ähnlich wie *FLARToolKit* ermöglicht diese Bibliothek die Registrierung anhand von Pattern-Markern. Im Gegensatz zu *FLARToolKit* werden jedoch die Markertypen *Binary Marker*, *Frame Marker*, *Split Marker* und *Datamatrix Marker* unterstützt. (Abb. 16) Der Algorithmus zur Erkennung der Marker basiert auf *ARToolKit*, *ARToolKitPlus* und *ARTag*. Dennoch handelt es sich nicht um eine Portierung sondern um eine Neuentwicklung. Der Vorteil gegenüber *FLARToolKit* liegt neben

[81] vgl. For Nerds Only: Cutsom FLAR Markers Explained. `http://www.squidder.com/2009/03/05/for-nerds-only-custom-flar-markers-explained/`, letzter Aufruf: 08.06.2010

der Vielzahl verschiedener Markersysteme an der robusteren Erkennung. Auch bei leichter Verdeckung können die Marker noch korrekt erkannt werden. Das ist durch die Verwendung einer Kantendetektion zum Auffinden der Marker möglich. (siehe 2.3.1.2) Ein weiterer Vorteil ist die automatische Anpassung an schwankende Lichtverhältnisse. Laut Entwicklerangaben ist dieses System das schnellste Marker Tracking System für die Plattform *Flash*.[82]

Abbildung 16.: Unterstützte Markersystem von flare*tracker
(Abb. aus: Augmented Reality für Flash. flare*tracker. `http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Atracker#`, letzter Aufruf: 03.06.2010)

*Flare*ntf* oder auch *Flash Augmented Reality Engine*Natural Feature Tracker* ist eine weitere kommerzielle Augmented Reality *Flash* Bibliothek der gleichen Entwickler. Diese Bibliothek setzt bei der Markererkennung und Registrierung auf einen so genannten *Natural Feature Tracker*. Hierbei können jegliche Bilder, die ausreichend markante Merkmale (Features) beinhalten, als Marker genutzt werden. Diese Features können zum Beispiel Kanten oder starke Konstrastbereiche sein. Das System analysiert ein Bild, dass in der Augmented Reality Anwendung als Marker dienen soll, auf markante Merkmale und speichert diese. In der Anwendung wird dann das Live-Videobild nach diesen Merkmalen durchsucht und anhand der Anordnung der gefundenen Merkmale die korrekte Registrierung durchgeführt. Der Vorteil dieser Technologie liegt darin, dass keine Pattern-Marker benutzt werden müssen. Vielmehr können ganze Bilder als Marker dienen. Für eine korrekte Registrierung müssen nicht zwingend alle Merkmale erkannt werden. Auch wenn das Bild teilweise verdeckt ist, kann es noch erkannt werden. Weiter ist diese Registrierungsmethode auch unemp-

[82] vgl. Augmented Reality für Flash. flare*tracker. `http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Atracker#`, letzter Aufruf: 03.06.2010

findlicher gegenüber Reflexionen, Unschärfe und extremer Winkel und Skalierung des Bildmarkers.[83] (Abb. 17)

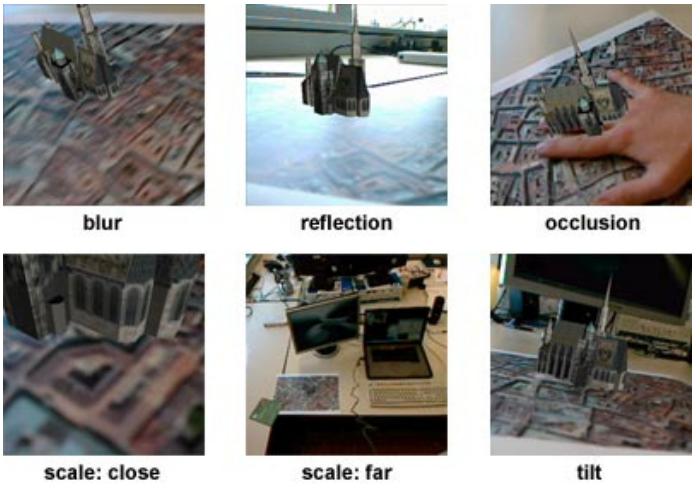


Abbildung 17.: Registrierung mittels Natural Feature Tracking mit flare*nft (Abbildung aus: Augmented Reality for Flash. flare*nft. `http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Anft`, letzter Aufruf: 03.06.2010)

3.2.2. Markerlose Regisitrierung in Augmented Reality RIAs

Bei der markerlosen Registrierung müssen Informationen aus der realen Szene als Marker verwendet werden. Es dürfen keine zusätzlichen Markierungen physikalisch in die reale Szene eingebracht werden. (siehe 2.3) Zum Zeitpunkt der Arbeit fand sich nur eine RIA, in der eine markerlose Registrierung umgesetzt wurde. (siehe 3.6.4) Dennoch ist das langfristige Ziel, Augmented Reality Anwendungen ohne Marker umzusetzen, da nur so eine nahtlose Verschmelzung von Realität und Virtualität stattfinden kann. Die Grundlage für eine markerlose Registrierung ist das Tracking von in der Szene bereits vorhandenen Objekten. Damit die für die Registrierung erforderlichen Daten gewonnen werden können, wird meist eine Kombination aus verschiedenen Tracking-Techniken verwendet. In den folgenden Abschnitten soll auf eine Auswahl an Tracking-Bibliotheken für *Flash* eingegangen werden.

[83] vgl. Augmented Reality for Flash. flare*nft. `http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Anft`, letzter Aufruf: 03.06.2010)

3.2.2.1. FaceIt Bibliothek

Die Bibliothek *FaceIt* von Benjamin Jung ist eine Umsetzung des CAMSHIFT-Algorithmus für *Flash*.[84] Der CAMSHIFT-Algortithmus (Continuously Adaptive Mean Shift) ist ein farbbasiertes Trackingverfahren mit dem sich Objekte in einer Videosequenz verfolgen lassen. Dem System muss zuvor die Farbverteilung bzw. das Farb-Histogramm des zu verfolgenden Objektes bekannt sein. In einem Suchfenster wird für jeden Pixel ermittelt, mit welcher Wahrscheinlichkeit dieser zum gesuchten Objekt gehört. Die Wahrscheinlichkeit wird anhand des zuvor gespeicherten Farbhistogramms bestimmt. Je größer die Summenhäufigkeit für den jeweiligen Farbwert im Referenzhistogramm ist, desto größer ist die Wahrscheinlichkeit, dass der aktuelle Farbwert und damit der Pixel zum gesuchten Objekt gehört. Mit dieser Methode erhält man ein Abbild der Wahrscheinlichkeiten jedes Pixels für jedes Videobild. Nun kann mit Hilfe des Mean-Shift Algorithmus die Position mit der höchsten Wahrscheinlichkeitsdichte ermittelt werden. Das entspricht dem Zentrum des zu verfolgenden Objektes. Das Suchfenster wird nun an diese Stelle verschoben. Anschließend wird durch Berechnung der Momente erster und zweiter Ordnung des Wahrscheinlichkeitsabbildes die Größe und Rotation des Objektes ermittelt und das Suchfenster daran angepasst.[85,86,87]

Der Algorithmus bietet ein robustes Trackingverfahren für Objekte, die in ihrer Textur annähernd die gleichen Farbwerte aufweisen. Dabei ist die Größe und Rotation im zweidimensionalen Raum anhand der Momente des Wahrscheinlichkeitsabbildes leicht zu berechnen. Schlechte Lichtverhältnisse können das Verfahren beeinträchtigen, da die nötigen Farbinformationen nicht vorhanden sind. Weiter führen Hintergründe mit der gleichen farblichen Zusammensetzung wie das zu verfolgende Objekt zu Problemen beim Tracking.

[84] FaceIt. `http://www.libspark.org/wiki/benj/FaceIt/en`, letzter Aufruf: 05.08.2010

[85] vgl. Computer Vision Face Tracking For Use in a Perceptual User Interface. 1 ff. `http://isa.umh.es/pfc/rmvision/opencvdocs/papers/camshift.pdf`, letzter Aufruf: 07.07.2010

[86] vgl. Jung, B.: Camshift: going to the source. `http://www.mukimuki.fr/flashblog/2009/06/18/camshift-going-to-the-source/`, letzter Aufruf: 07.07.2010

[87] vgl. Moment (Bildverarbeitung). `http://de.wikipedia.org/wiki/Moment_(Bildverarbeitung)`, letzter Aufruf: 07.07.2010

3.2.2.2. ASSURF Bilbiothek

Die Bibliothek *ASSURF* von Eugene Zatepyakin ist eine Umsetzung des SURF Algorithmus für *Flash*. Der Surf Algorithmus (Speeded Up Robust Features) ermöglichst das Finden und Verfolgen von Bildern anhand von Merkmalen. Diese Merkmale, auch Features genannt, sind markante Punkte in einem Bild wie zum Beispiel Kanten. Ein Referenzbild wird auf diese Merkmale untersucht und die Ergebnisse als Referenzpunkte gespeichert. In dem zu untersuchenden Bild wird nun nach diesen gespeicherten Merkmalen gesucht. Wird eine festgelegte Menge an korrespondierenden Merkmalen gefunden gilt das Referenzbild als erkannt. Die Erkennung ist dabei skalierungs- und rotationsunabhängig, das heißt, das zu findende Bild muss in seiner Größe und Lage nicht dem Referenzbild entsprechen, um erfolgreich gefunden zu werden. Durch die Verwendung von mehreren Merkmalen ist dieses Verfahren auch unempfindlich gegenüber Verdeckung, solange noch genügend Merkmale unverdeckt bleiben. Mit *ASSURF* lässt sich ein markerbasiertes System wie der *flare*nft* (siehe 3.2.1.2) umsetzen. Auch eine markerlose Augmented Reality Anwendung ist denkbar. So können mit diesem Algorithmus markierte Objekte verfolgt werden, indem man die Bewegung der Merkmalspunkte in einer Bildfolge analysiert.[88,89]

3.2.2.3. Marilena und Haar Cascades Detector Bibliothek

Die Bibliothek *Marilena* ist eine Umsetzung des Viola-Jones-Algorithmus für *Flash*.[90] Auch Eugene Zatepyakin hat mit der *Haar Cascades Detecor* Bibliothek den Algorithmus für *Flash* umgesetzt. Laut Angaben des Autors ist Bibliothek *Haar Cascades Detecor* die schnellste Umsetzung für *Flash*.[91]

Der Algorithmus wurde zum Auffinden von Gesichtern in Bildern entwickelt. Aber auch andere Objekte können gefunden werden. Der Algorithmus arbeitet mit einem Suchfenster, das über das Bild geschoben wird. Anhand der Helligkeitswerte des Bildausschnitts werden sogenannte Blockmerkmale errechnet. Jeder Bildaus-

[88] vgl. Zatepyakin, E.: ASSURF. SURF library for Adobe Flash Platform. `http://code.google.com/p/in-spirit/wiki/ASSURF`, letzter Aufruf: 08.07.2010

[89] vgl. BAY/ESS/TUYTELAARS et al. 2008, 1 ff.

[90] Marilena. `http://www.libspark.org/wiki/mash/Marilena`, letzter Aufruf: 08.07.2010

[91] Zatepyakin, E.: Haar Cascades Detector. `http://code.google.com/p/in-spirit/wiki/HaarCascadesDetector`, letzter Aufruf:05.08.2010

schnitt durchläuft dann eine Kaskade. Diese besteht aus verschiedenen Klassifikatoren, die jeweils eine positive oder negative Antwort geben können. Die Anzahl der auszuwertenden Merkmale je Klassifikator wird dabei immer größer und damit auch komplexer. Erst wenn alle Klassifikatoren positiv durchlaufen wurden, wird der Bildausschnitt als Gesicht erkannt. Das Verfahren ist skalierungs- aber nicht rotationsunabhängig. Sobald sich der Kopf zu einem gewissen Grad neigt, wird dieser nicht mehr von dem Detektor erkannt. Ein Video von Adam Harv veranschaulicht diesen Algorithmus.[92]

Die Klassifikatoren, die bei einer Entscheidungsfindung durchlaufen werden, sind in einer sogenannten Haar-Kaskade gespeichert. Die Kaskade ist spezifisch für das zu findende Objekt. Es muss also für das Finden des Gesichts eine andere Haar-Kaskade verwendet werden, als für das Finden von Augen oder Mund. Um solch eine Kaskade zu erstellen, muss eine Anzahl an Positiv- und Negativbeispielen für das gesuchte Objekt gefunden werden. In den Positivbeispielen muss das gesuchte Objekt markiert werden. Die Negativbeispiele dürfen das Objekt hingegen nicht enthalten. Eine Software errechnet dann aus diesen Beispielen die Haar-Kaskade.[93]

3.3. 3D Engines für Flash

Nach Azuma ist für die Umsetzung von Augmented Reality die dreidimensionale Darstellung virtueller Objekte notwendig. (siehe 2) Um 3D-Modelle in *Flash* darstellen zu können, benötigt man eine 3D-Engine in Form einer externen Programmbibliothek. Zum Zeitpunkt dieser Arbeit haben sich die Bibliotheken *Papervision3D*, *Away3D*, *Away3D lite* und *Sandy3D* als die verbreitetsten herausgestellt. Alle Bibliotheken sind Open-Source und für den privaten und kommerziellen Einsatz kostenlos. Im nächsten Abschnitt soll näher auf die verschiedenen Engines eingegangen werden.

[92] Haar, Adam: OpenCV Face Detection Visualized. `http://vimeo.com/12774628`, letzter Aufruf: 08.07.2010

[93] vgl. VIOLA/JONES 2001, 1 ff.

3.3.1. Vergleich der Engines Papervision3D, Away3D, Away3D lite und Sandy 3D

3.3.1.1. Vergleich der Performance

An dieser Stelle soll beispielhaft die Anzeige eines 3D-Modells mit den verschiedenen Engines in einer *Flash* Anwendung gezeigt werden. Dieser Test soll die Performance der Engines untereinander vergleichen. Dafür wird jeweils ein 3D-Modell im *Collada*-Dateiformat[94] importiert und um seine eigene Achse rotiert.

Bei diesem Test nimmt *Away3D lite* eine gewisse Sonderstellung ein, da diese Bibliothek auf hohe Geschwindigkeit und geringen Speicherbedarf optimiert wurde. Laut Entwickler ist *Away3D lite* bis zu vier mal schneller als andere Engines. Ein Grund hierfür ist die Verwendung von nativen 3D Funktionen, die in der Version 10 des *Flash* Players implementiert wurden. Eine Verwendung mit älteren Versionen des *Flash* Players ist folglich nicht möglich. Weiter soll die Dateigröße im Vergleich zu den anderen Engines wesentlich geringer sein. Dies liegt daran, dass nur ein Teil der Funktionen aus *Away3D* implementiert wurden.[95]

Tabelle 3.1 zeigt die Ergebnisse des Versuchs. Der Test wurde auf einem *Macbook Pro* mit 2.4 GHz *Intel Core 2 Duo* Prozessor, 4 GB DDR3 Speicher und der Standalone *Flash* Player Version 10.1.53.64 durchgeführt. Die verwendeten Bibliotheken wurden jeweils in der aktuellsten Version (Stand Juni 2010) verwendet. Es wurde die Bildrate pro Sekunde, der benutzte Arbeitsspeicher sowie die Dateigröße der *swf*-Datei ermittelt und der durchschnittliche Zeitwert gemessen, der für das Rendern eines Frames benötigt wurde. Der Test soll einen Anhaltspunkt für die Wahl einer 3D-Engine für *Flash* Augmented Reality Anwendungen geben.

In den Punkten Geschwindigkeit und Speicherauslastung liegt *Away3D lite* klar vor den anderen Engines. *Papervision3D*, *Away3D lite* und *Sandy 3D* bieten in etwa die gleiche Performance beim Rendern, nur *Away3D* benötigt deutlich mehr Rechen-

[94] Collada ist ein XML basiertes, freies Dateiformat für den Austausch von 3D-Modellen zwischen verschiedenen 3D Grafik-Programmen. (vgl. COLLADA. `https://collada.org/mediawiki/index.php/COLLADA`, letzter Aufruf: 21.06.2010)

[95] vgl. Away3D Lite v1.0: Fastest and Smallest 3D Engine in Flash. September 2009, `http://away3d.com/away3d-lite-v1-0-fastest-and-smallest-3d-engine-in-flash`, letzter Aufruf: 10.06.2010

	Papervision3D	Away3D	Away3D lite	Sandy 3D
Version	2.1.932	3.5.0	1.0.0	3.1.2
Bilder pro Sekunde	60	50	60	60
Berechnungszeit pro Frame	3 MS	10 MS	2 MS	5 MS
Benutzer Speicher	14 MB	22 MB	10 MB	21 MB
Dateigröße	74 KB	152 KB	160 KB	61KB

Tabelle 3.1.: Performance Vergleich zwischen den Open Source 3D-Engines in *Flash*

zeit.[96] Die erwartete geringe Dateigröße von *Away3D lite* hat sich in diesem Test nicht bestätigt. Um aber die Eignung einer Engine für eine Augmented Reality Anwendung beurteilen zu können, müssen auch die darstellungsspezifischen Funktionen gegenüber gestellt werden.

3.3.1.2. Vergleich der darstellungsspezifischen Funktionen

Ein vollständiger Vergleich aller Funktionen der Engines ist im Rahmen dieser Arbeit nicht möglich. Vielmehr sollen die Funktionen verglichen werden, die Einfluss auf die realistische Darstellung der dreidimensionalen Objekte haben und damit Lösungsmöglichkeiten für darstellungsspezifische Probleme bieten. (siehe 2.4) Die Bewegungsfaktoren werden durch alle Engines abgedeckt. Die physiologischen und binokularen Faktoren haben bei Augmented Reality Anwendungen in RIAs keine Bedeutung, da es als Fokusebene nur den Bildschirm gibt und die stereoskopische Darstellung nicht Gegenstand dieser Untersuchung sein soll. Daraus ergeben sich als darstellungsspezifische Faktoren die Darstellung von Licht und Schatten und die korrekte Darstellung von Verdeckung. Die korrekte Darstellung von Verdeckung ist nicht engine-spezifisch und wird daher unter Abschnitt 3.4 gesondert betrachet.

An dieser Stelle soll die Darstellung von Licht und Schatten untersucht werden. Dazu zählt das Setzen von Lichtquellen, die daraus resultierenden Effekte für die Textur eines 3D-Modells und die korrekte Darstellung von Schatten. In Tabelle

[96] Anmerkung zu Sandy 3D: Die falsche Darstellung ist vermutlich auf das importierte Modell zurück zu führen. Für diesen Test ist dies jedoch unerheblich, da alle Elemente des Modells geladen und dargestellt wurden. Somit ist die Anzahl der zu berechnenden Polygonen in allen Anwendungen gleich.

3.2 ist ein Vergleich der implementierten Lichtarten, Schatten und Shader[97] aufgeführt.[98,99,100,101]

	Papervision3D	Away3D	Away3D lite	Sandy 3D
Version	2.1.932	3.5.0	1.0.0	3.1.2
Lichtquellen	Point	Ambient Directional Point	—	Directional
Shader	Flat Phong Environment Gouraud Cell	Flat Phong Environment DOT3	—	Flat Phong Environment Gouraud Cell
Schatten	externe Klasse[102]	Simple Shadow[103]	—	—

Tabelle 3.2.: Licht-, Schatten- und Shading-Funktionen der 3D-Engines

In der Gegenüberstellung wird deutlich, wie sich der Funktionsumfang auf die zuvor gezeigte Performance auswirkt. *Away3D lite* ist nur mit den grundlegenden Funktionen einer 3D-Engine ausgestattet und bietet keine Lichtquellen, Schattenbildung und Shader. Die anderen Engines besitzen in Bezug auf Lichtquelle und Shader annähernd den gleichen Funktionsumfang. Hervorzuheben ist der in *Away3D* implementierte DOT3 Shader, der das sogenannte Normal-Mapping erlaubt. Dadurch kann ein 3D-Modell mit wenigen Polygonen vergleichsweise hoch aufgelöst wirken, was zu

[97] Die Darstellung der Textur eines 3D-Modells bei Lichteinfall wird von einem sogenannten Shader berechnet. Dabei gibt es verschiedene Algorithmen, die sich vor allem in der Qualität der Darstellung und im Rechenaufwand unterscheiden.

[98] vgl. Papervision3D API Documentatio. `http://www.papervision3d.org/docs/as3/index.html`, letzter Aufruf: 21.06.2010

[99] vgl. Away3D FP10 V3.5.0 API Documentation. `http://away3d.com/livedocs/3.5.0_lib/`, letzter Aufruf: 21.06.2010

[100] vgl. Away3DLite API Reference. `http://away3d.com/livedocs/fp10/Away3DLite/`, letzter Aufruf:21.06.2010

[101] vgl. Sandy 3D Engine, 3.1.2 `http://sandy.googlecode.com/svn/trunk/sandy/as3/tags/3.1.2/docs/index.html`, letzter Aufruf: 21.06.2010

[102] Eine Erweiterung für Papervision erlaubt die dynamische Darstellung von Schatten auf einer Ebene. (siehe CASTING SHADOWS IN PAPERVISION - REDUX.`http://blog.zupko.info/?p=146`, letzter Aufruf: 21.06.2010

[103] Erstellt statische Schatten auf einer Ebene. Ein dynamische Schattendarstellung, für sich bewegende Objekte, ist theoretisch möglich, ist jedoch laut Entwickler zu Berechnungsintensiv. (vgl. How to Use SimpleShadow. `http://away3d.com/tutorials/how-to-use-simpleshadow`, letzter Aufruf: 21.06.2010)

einer realistischeren Darstellung führt.[104] Diese Aufstellung ist rein quantitativer Natur und bewertet nicht die Rendering-Qualität der jeweiligen Engine.

3.4. Korrekte Darstellung von Verdeckung

Unter 2.4.2 wurden verschiedene Lösungsansätze für die Verdeckungsdarstellungsproblematik aufgezeigt. Es erfolgte eine Unterscheidung in manuelle, halbautomatische und automatische Verfahren. Mit *Flash* lassen sich die manuellen und halbautomatischen Verfahren umsetzen.

Die manuelle Erstellung von zweidimensionalen Masken kann in *Flash* durch die *mask*-Eigenschaft der *DisplayObject*-Klasse umgesetzt werden. Dabei wird ein zweidimensionales Displayobjekt wie eine *Sprite*- oder *MovieClip*-Instanz erstellt und als Maskenebene angegeben. Die Teile des zu maskierenden Displayobjekts, die von der Maskenebene verdeckt werden, werden nicht angezeigt. Die Maske selbst ist nicht sichtbar. Die Maske kann in ihrer Position, Größe und Form sowohl statisch als auch dynamisch sein. [105]

Die dynamische Verdeckung vor statischen Hintergründen lässt sich mit einem *Bitmap Filter* realisieren. Diese Filter können mit dem Tool *Adobe Pixel Bender*[106] programmiert werden. Mit Hilfe des Filters wird eine dynamische Maske erstellt, die nur an den Stellen verdeckt, an denen das aktuelle Videobild von dem zuvor gespeicherten Hintergrundbild verschieden ist. Dazu wird jeder Pixelwert des Hintergrundbildes von dem aktuellen Videobild subtrahiert. Wenn das Ergebnis gegen Null geht, also das Videobild gleich dem Hintergrund ist, wird der Alpha-Wert des jeweiligen Pixels auf durchsichtig gesetzt. Um leichte Schwankungen der Pixelwerte zum Beispiel durch Beleuchtungsunterschiede zu berücksichtigen, kann ein Schwellwert eingeführt werden. Wenn das Ergebnis der Subtraktion unterhalb des Schwellwertes liegt, wird es als Hintergrund angesehen.

[104] siehe: Normal Mapping. `http://de.wikipedia.org/wiki/Normal_Mapping`, letzter Aufruf: 21.06.2010

[105] vgl. ActionScript 3.0 Language and Components Reference. Display Object. `http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/DisplayObject.html#mask`, letzter Aufruf: 29.06.2010

[106] Pixel Bender Technology Center. `http://www.adobe.com/devnet/pixelbender/`, letzter Aufruf: 29.06.2010

In der in Abbildung 18 gezeigten Anwendung wurde versucht, dieses Konzept umzusetzen. Der grüne Kreis repräsentiert dabei ein virtuelles Objekt, das zwischen einem dynamischen Vordergrund und einem statischen Hintergrund liegt. In dem Beispielprogramm wurde vor der Subtraktion der Pixel ein Weichzeichnungsfilter eingesetzt, um kleinere Störungen zu beseitigen. Weiter wurde die Subtraktion von Videobild und Hintergrund im HSV-Farbraum durchgeführt, um die Einflüsse von schwankenden Lichtverhältnissen zu minimieren. Der kommentierte Quelltext des *Pixel Bender* Filters befindet sich unter Anlage B.

Abbildung 18.: Beispielanwendung zur Dynamischen Verdeckung vor statischen Hintergrund.

Die Verdeckung durch vorkonstruierte 3D-Modelle ist mit Chroma Keying möglich. Ein zuvor konstruiertes 3D-Modell wird mit 100% grün texturiert. Für die Texturierung ist jede Farbe geeignet, die möglichst nicht im restlichen Bild vorkommt. Das Modell wird nun in Lage und Position mit dem korrespondierenden realen Objekt in Übereinstimmung gebracht. Damit dieses Modell in der Augmented Reality Szene nicht sichtbar ist, wird der Alpha-Wert von allen Pixeln auf durchsichtig gesetzt, die einen Farbwert von 100% grün aufweisen. Dies ist mit der Klasse *ColorMatrixFilter* möglich.[107,108]

Ein Probleme dieser Methode ist zum einen, dass die grünen Umrisskanten des Maskenobjektes eventuell leicht zu sehen sind. Zum anderen wird durch den *Color Matrix Filter* auch die Darstellung des Grünanteils anderer Objekte verändert. Letzteres Problem lässt sich durch die Verwendung eines eigenen Filters vermeiden. Der Filter überprüft dabei jeden Pixel im Bild. Wenn der Farbwert des Pixels einen genügend hohen Grünanteil aufweist und gleichzeitig die Rot- und Blauwerte genügend gering sind, wird der Pixel auf durchsichtig gesetzt. Abbildung 19 veranschaulicht die genannten Probleme. Der grüne Würfel repräsentiert das vorkonstruierte 3D-Modell,

[107] ActionScript 3.0 Language and Components Reference. ColorMatrixFilter. `http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/filters/ColorMatrixFilter.html`, letzter Aufruf: 29.06.2010

[108] How to make only „The Hole“using Papervision3D. `http://saqoosha.net/en/2009/01/08/1676/`, letzter Aufru:29.06.2010

das in einer späteren Anwendung mit einem realen Objekt in Lage, Position und Größe in Übereinstimmung gebracht wird. Der rote Würfel repräsentiert das virtuelle Objekt, das verdeckt werden soll. Abbildung 19 (a), (b) zeigt die Verwendung des *Color Matrix Filters*. Bei Abbildung 19 (b) ist deutlich die ungewollte Farbänderung des dritten Würfels zu sehen. Abbildung 19 (c), (d) zeigt die Verwendung eines *Pixel Bender* Filters mit dem zuvor erläuterten Verfahren. Hierbei tritt keine Farbänderung des dritten Würfels auf. Der kommentierte Quelltext des *Pixel Bender* Filters befindet sich unter Anhang C.

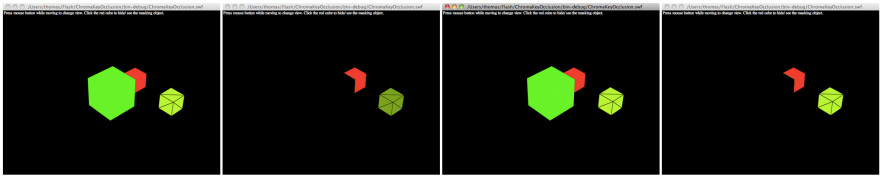


Abbildung 19.: (a), (b): Verwendung des Color Matrix Filter; (c), (d): Verwendung des Pixel Bender Filters

3.5. Fazit

In den voran gegangenen Abschnitten wurden die Möglichkeiten von Augmented Reality in RIAs gezeigt. Nach eigener Einschätzung liegt das Potential vor allem im Bereich des E-Commerce, in der online Produktdarstellung und -werbung. Durch die Umsetzung als RIA ist eine nahtlose Integration in Webseiten möglich. Weiterhin bietet Augmented Reality im Kontext eines Onlineshops einen Mehrwert für den Kunden, der eine höhere Konversion zur Folge haben kann.

Mit dem *FLARToolKit* ist die Erstellung von markerbasierten Augmented Reality Anwendungen in *Flash* mit geringem Aufwand möglich. Dabei lässt sich mit der Verwendung des Frameworks *FLARManager* der zu schreibende Quellcode im Vergleich zu *FLARToolKit* weiter reduzieren. Der *ARToolKit Marker Generator Online* ermöglicht es, ohne Programmierkenntisse individuelle Marker und die dazugehörigen Pattern-Dateien zu erstellen. Für nicht kommerzielle, makerbasierte Augmented Reality RIAs bieten diese Bibliotheken eine gute Lösung für die Registrierungsproblematik. Für den kommerziellen Einsatz sind alle auf dem *ARToolKit* basierenden Programme lizenzpflichtig, so dass nach eigener Einschätzung die Verwendung von

*flare*tracker* bzw. *flare*nft* gegenüber *FLARToolKit* vorzuziehen sind. Es werden eine größere Anzahl an möglichen Markerarten unterstützt und zudem ein robusteres und schnelleres Trackingverfahren angewendet. Weiter ist mit *flare*nft* die Verwendung von reinen Bildern als Marker zum Beispiel in Form einer Printanzeige möglich. Der Entwickler von *FLARManager* hat weiterhin eine Unterstützung dieser beiden Bibliotheken in der nächsten Programmversion angekündigt, was eine einfache Verwendung verschiedener 3D-Engines für *Flash* mit *flare*tracker* bzw. *flare*nft* ermöglicht.[109]

Für die markerlose Registrierung gibt es bis auf die Face-Tracking Technologie von *Total Immersion* bisher keine fertigen Lösungen. Vielmehr muss mittels Kombination verschiedener Trackingtechniken versucht werden, eine der jeweiligen Anwendung entsprechende Registrierung umzusetzen.

Bei den verschiedenen 3D-Engines für *Flash* scheint *Away3D lite* durch die im Vergleich zu den anderen 3D-Engines bessere Performance und durch den geringen Speicherbedarf besonders interessant für den Einsatz in Augmented Reality Anwendung zu sein. Es besteht die Möglichkeit, komplexe texturierte 3D-Modelle zu laden. Dabei ist das Setzen von Lichtquellen und die Berechnung von Schatten sowie die Verwendung von besonderen Shading-Algorithmen mit *Away3D lite* nicht möglich. *Away3D lite* sollte vorrangig eingesetzt werden, wenn die Augmented Reality Anwendung auf die Darstellung von Licht- und Schatteneffekten und auf die Verwendung spezieller Shader verzichten kann. Bei Augmented Reality Anwendungen, in denen solche Darstellungen gewünscht sind, ist nach eigener Einschätzung *Papervision3D* bzw. *Sandy 3D* gegnüber *Away3D* vorzuziehen. Für optisch anspruchsvolle Anwendungen bietet *Away3D* aufgrund der verschiedenen Lichtquellen und dem DOT3-Shader die beste Funktionalität.

Abschließend muss erwähnt werden, dass die oben genannten Vor- und Nachteile den Entwicklungsstand zum Zeitpunkt dieser Arbeit (Juli 2010) widerspiegeln. Alle genannten Bibliotheken werden sehr schnell von einer großen Entwicklergemeinde weiterentwickelt. Dadurch können sich die aufgezeigten Funktionen und Performance-Werte innerhalb weniger Monate ändern.

[109] vgl. Socolofsky, Eric: FLARManager for flare/NFT. `http://words.transmote.com/wp/20100520/flarmanager-for-flarenft/`, letzter Aufruf: 08.07.2010

3.6. Beispiele für Augmented Reality RIAs

3.6.1. IKEA - 30 Jahre BILLY

Zum 30 jährigen Jubiläum des *IKEA BILLY* Regals gibt es auf der *IKEA*-Webseite eine Augmented Reality Anwendung, mit der man ein virtuelles Regal im Live-Bild der Webcam platzieren kann. Die Anwendung ist auf Basis von *Flash* und verwendet eine markerbasierte Registrierung mit einem quadratischen Pattern-Marker. Nachdem der Benutzer den Marker ausgedruckt hat, kann er diesen frei im Raum platzieren. Im Live-Bild der Webcam wird nun das Möbelstück an der Position des Markers dargestellt. Dieses Beispiel zeigt, wie eine zukünftige E-Commerce Augmented Reality Anwendung aussehen könnte.[110]

Abbildung 20.: Ikea Augmented Reality Anwendung
(Abb. aus: IKEA | BILLY 30 år. `http://www.ikea.com/ms/sv_SE/kampanj/fy10/billy30/`, letzter Aufruf: 02.06.2010)

[110] siehe: IKEA | BILLY 30 år. `http://www.ikea.com/ms/sv_SE/kampanj/fy10/billy30/`, letzter Aufruf: 02.06.2010

3.6.2. USPS Packet Simulator

Eine weitere Augmented Reality RIA Anwendung ist der *Virtual Box Simulator* des *United States Postal Service*. Hier kann der Benutzer ausprobieren, ob die Gegenstände, die er verschicken möchte, in den Versandkarton des Postdienstleisters passen. Bei dieser Anwendung ist eine korrekte Größendarstellung des virtuellen Objektes von besonderer Bedeutung. Aus diesem Grund wurde für die Registrierung auf quadratische Pattern-Marker zurückgegriffen. Damit ist gewährleistet, dass die Größe des virtuellen Objektes mit der des realen Versandkartons übereinstimmt.[111]

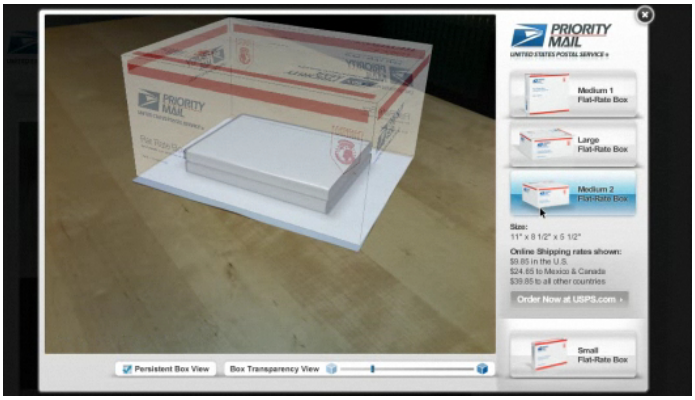


Abbildung 21.: Virutal Box Simulator des USPS
(Abb. aus: USPS Priority Mail - Virtual Box Simulator. `https://www.prioritymail.com/simulator.asp`, letzter Aufruf: 02.06.2010)

3.6.3. Mini Print Campagne

Die Printkampagne für den *Mini Cabrio* wurde durch eine Augmented Reality Anwendung erweitert. Dabei fungiert die Printanzeige als Marker für eine Augmented Reality Anwendung auf der Produktwebseite. Wenn der Benutzer die Printanzeige in die Webcam zeigt, erscheint ein 3D-Modell des *Mini Cabrios*. Durch Bewegen der Printanzeige kann das 3D-Modell von allen Seiten betrachtet werden. Der potentielle Kunde bekommt nicht nur einen besseren Eindruck vom Produkt, sondern wird gleichzeitig auf die Webpräsenz des Unternehmens geleitet. Es findet gewissermaßen eine Verschmelzung von Print- und Onlinemedium statt. Ein weiterer positiver

[111] siehe: USPS Priority Mail - Virtual Box Simulator. `https://www.prioritymail.com/simulator.asp`, letzter Aufruf: 02.06.2010)

Nebeneffekt dieser Kampagne war die große mediale Aufmerksamkeit, die durch die Augmented Reality Anwendung erzeugt wurde.[112]

Abbildung 22.: Augmented Reality Printkampagne für das MINI Cabrio
(Abb. aus: MINI ||. `http://www.flickr.com/photos/metaio/4473598564/`, letzter Aufruf: 02.06.2010)

3.6.4. Total Immersion D'Fusion

Das Unternehmen *Total Immersion* stellt Augmented Reality Anwendungen in allen Bereichen her. Ein Beispiel für markerlose Augmented Reality ist dabei die Anwendung auf der Webseite `http://www.weareautobots.com`. Im Live-Videobild der Webcam wird auf den Kopf des Benutzers eine virtuelle, dredimensionale Maske aus dem Film *Transformers* projiziert. Die Registrierung erfolgt dabei anhand eines Kopf-Tracking Algorithmus, der die Position, Bewegung und Rotation des Kopfes im Videobild erfasst. Die Registrierung wird ohne zusätzliche Marker durchgeführt und kommt auch ohne eine manuelle Initialisierung des Systems aus. Die Anwendung greift auf eine proprietäres Browser-Plugin des Unternehmens zurück, das aber auf der Basis von *Adobe Flash* arbeitet. Es ist eine Version des Systems angekün-

[112] World Premiere: The Augmented Reality Ad For MINI. `http://www.metaio.de/fileadmin/homepage/Dokumente/English/Best_Practice_MINI.pdf`, letzter Aufruf: 02.06.2010

digt, das mit dem normalen *Flash*-Plugins ausgeführt werden kann und somit keine zusätzliche Plugin-Installation erfordert.[113,114]

[113] vgl. Total Immersion. FAQ. http://www.t-immersion.com/en,faq,825.html, letzter Aufruf: 05.06.2010

[114] vgl. Total Immersion. Datasheet @Home Applications. http://www.t-immersion.com/lc-content/uploads/2009/11/datasheethome.pdf, letzter Aufruf: 05.07.2010

4. Konzipierung einer Augmented Reality Anwendung mit Adobe Flash

4.1. Beschreibung der Anwendung

Mit dieser Augmented Reality Anwendung für *Adobe Flash* soll der Benutzer eine virtuelle Brillenanprobe an einem Computer durchführen können. Mittels einer Webcam wird der vor dem Computer sitzende Benutzer gefilmt und das Live-Video Bild auf dem Bildschirm angezeigt. Der Benutzer wird nun aufgefordert, seinen Kopf im Videobild mit der Maus zu markieren. Dazu zieht er mit der Maus eine rechteckige Markierung um seinen Kopf. Diese dient zur Initialisierung des Tracking-Algorithmus. Anschließend wird ein 3D-Modell einer Brille geladen und auf den Kopf des Benutzers gelegt. Dabei folgt dieses Modell in Größe, Rotation und Neigung den Kopfbewegungen. Durch die Anwendung bekommt der Benutzer einen realistischen Eindruck von dem gewählten Brillenmodell. Eine Anwendung, in der ein ähnliches Prinzip umgesetzt wurde, ist der *Ray Ban Virtual Mirror*. Hier muss jedoch ein Programm auf dem Rechner installiert werden. Dieses Programm ist nur für das Betriebssystem *Microsoft Windows* erhältlich.[115] Um eine Plattformunabhängigkeit zu gewährleisten, soll eine ähnliche Anwendung wie der *Ray Ban Virtual Mirror* mit *Adobe Flash* umgesetzt werden. Das so erstellte Programm kann nahtlos in Webseiten bzw. Online-Shops integriert werden, was einen direkten Kauf des Brillenmodells über die Webseite des Händlers ermöglicht.

4.1.1. Anforderungen

Der Kernpunkt der Anwendung bildet ein Trackingsystem, das die Bewegung des Kopfes in sechs Richtungen verfolgt. (Abb. 23) Dabei soll ein maximaler Dreh- und Senkwinkel von ca. ± 30 Grad sowie einen Neigungswinkel von ca. ± 45 Grad des Kopfes verfolgt werden. Größere Winkel sind nicht erforderlich, da der Benutzer sonst den vor ihm liegenden Bildschirm nicht mehr richtig erkennen könnte. Der Abstand des Kopfes von der Webcam soll zwischen ca. 0,2 und 0,7 Meter betragen.

[115] Ray Ban. Virtual Mirror. `http://www.ray-ban.com/germany/science/virtual-mirror`, letzter Aufruf: 14.07.2010

Das 3D-Modell soll den Bewegungen des Kopfes folgen. Dabei dürfen die Teile des Modells nicht sichtbar sein, die von dem Kopf verdeckt werden würden.

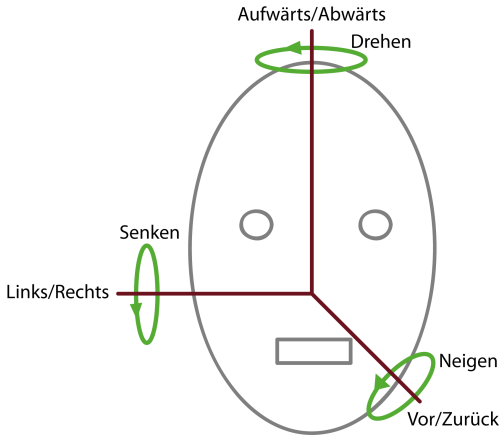


Abbildung 23.: Horizontale und vertikale Dreh- und Neigungsrichtung des Kopfes

4.2. Tracking

4.2.1. Gesicht-, Augen- und Mund-Tracking mit CAMSHIFT- und Viola-Jones-Algorithmus

Die zweidimensionale Lage und Größe des Kopfes wird mit dem CAMSHIFT-Algorithmus ermittelt. Dafür kann die Bibliothek *FaceIt* verwendet werden. (siehe 3.2.2.1) Der CAMSHIFT-Tracker muss durch Markieren des Gesichtes im Videobild initialisiert werden. Anschließend verfolgt der Tracker das markierte Gesicht im Videobild und ermittelt die Höhe, Breite und Lage des Gesichtes. Die errechnete Lage entspricht dabei dem Mittelpunkt des Gesichtes. Der Mittelpunkt bleibt auch während einer Drehung und Senkung des Kopfes konstant.

Für das Finden und Verfolgen von Augen und Mund eignet sich der Viola-Jones Algorithmus. (siehe 3.2.2.3) Dafür kann die Bibliothek *Haar Cascade Detector* von Eugene Zatepyakin verwendet werden.[116] Man benötigt jeweils eine Haar-Kaskade für das Auge und für den Mund. In ersten Tests erwiesen sich die Kaskaden von Shameem Hameed für das Finden der Augen und von Modesto Castrillon-Santana für

[116] Zatepyakin, Eugene: Haar Cascade Detector. `http://code.google.com/p/in-spirit/wiki/HaarCascadesDetector`, letzter Aufruf: 03.08.2010

das Finden des Mundes als zuverlässig.[117,118] Um eine bessere Performance zu erreichen, werden die jeweiligen Suchbereiche für das linke, das rechte Auge und den Mund eingegrenzt. Die Größe des menschlichen Kopfes und die Anordnung von Augen und Mund stehen in einem festen Verhältnis zueinander, so dass die Suchbereiche anhand der Ergebnisse des CAMSHIFT-Algorithmus berechnet werden können. Die Breite und Höhe des Kopfes haben ein Verhältnis von 2:3 (Abb. 24). Die roten Rechtecke repräsentieren die berechneten Suchfenster.

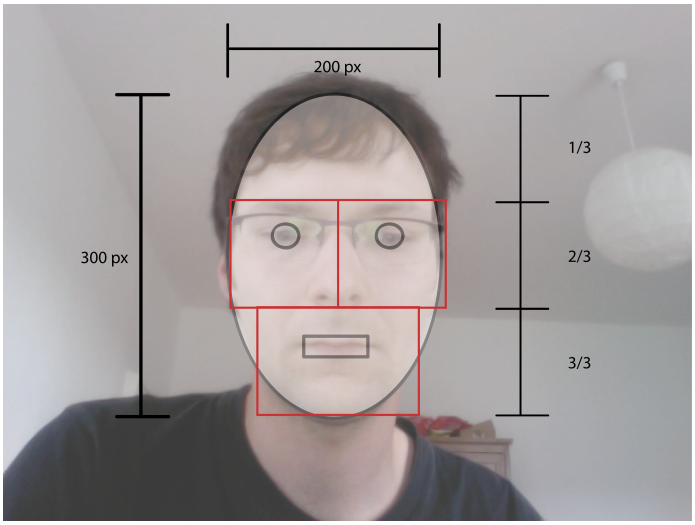


Abbildung 24.: Verhältnis Höhe und Breite des Kopfes und daraus errechnete Suchfenster

Daraus ergeben sich folgende Formeln für die Berechnung der Suchbereiche. Die Werte x und y repräsentieren dabei die linke obere Ecke und die Werte w und h die Breite und Höhe des Suchfenster. Die Werte $Face_x$, $Face_y$, $Face_{width}$ und $Face_{height}$ sind die Ergebnisse des CAMSHIFT-Algorithmus.

Linkes Auge:

$$x = Face_x - \frac{Face_{width}}{2} \qquad w = \frac{Face_{width}}{2}$$
$$y = Face_y - \frac{Face_{width}}{4} \qquad h = \frac{Face_{width}}{2}$$

[117] Hameed, Shameem: HAARcascade for eyes. `http://www-personal.umich.edu/~shameem/haarcascade_eye.html`, letzter Aufruf:03.08.2010

[118] Castrillón-Santana, Modesto:Face Features Detectors. `ftp://mozart.dis.ulpgc.es/pub/Software/HaarClassifiers/FaceFeaturesDetectors.zip`, letzter Aufruf:03.08.2010

Beim rechten Auge ändert sich im Vergleich zum linken Auge lediglich der x-Wert:

$$x = Face_x$$

Mund:

$$x = Face_x - \frac{Face_{width}}{4} \qquad w = Face_{width} * \frac{3}{4}$$
$$y = Face_y + \frac{Face_{width}}{4} \qquad h = \frac{Face_{width}}{2}$$

Wenn das gesuchte Objekt gefunden wurde, wird der Mittelpunkt des jeweiligen Suchfenster an den Mittelpunkt des gefundenen Objekts verschoben. Dadurch wird das Tracking auch gewährleistet, wenn durch Drehung, Neigung oder Senkung des Kopfes die festen Seitenverhältnisse des Gesichts und damit eine korrekte Positionierung der Suchfenster nicht mehr gegeben ist. Wenn ein Objekt im seinem Suchbereich nicht mehr gefunden werden kann, wird das jeweilige Suchfenster nach den oben genannten Berechnungen neu initialisiert.

Das beschriebene Trackingverfahren wurde in der Anwendung *TrackFace* umgesetzt. Um den Tracker zu initialisieren muss mit der Maus eine Markierung gezogen werden, die das Gesicht zwischen Augenbrauen und Kinnspitze beinhaltet.

4.2.2. Errechnung der Dreh-, Senkungs- und Neigunswinkel

Um das 3D-Modell der Brille den Kopfbewegungen folgen zu lassen, ist es notwendig, neben der Position auch die Drehung, Neigung und Senkung des Kopfes zu ermitteln. (Abb. 23) Für die Neigung des Kopfes wird die Position der Augen auf der y-Achse verglichen.

$$Neigung = leftEye_y - rightEye_y$$

Für die Ermittlung der Drehung des Kopfes wird der Abstand der Augen oder des Mundes zum Mittelpunkt des Gesichts in Richung der x-Achse und für die Ermittlung von Heben und Senken des Kopfes in Richtung der y-Achse verwendet. Durch die verschiedenen Berechnungsmöglichkeiten kann bei einem Fehlschlagen des Trackings von einem Objekt immer noch auf eine andere Berechnungsmethode zurückgegriffen werden. Nachfolgend ist jeweils die Formel für die Berechnung mit der Position des linken und rechten Auges sowie mit der Position des Mundes dargestellt.

$$Drehung = (Face_x - leftEye_x - \frac{Face_{width}}{5}$$
$$Drehung = Face_x - rightEye_x + \frac{Face_{width}}{5}$$
$$Drehung = (Face_x - Mouth_x) * 0.7$$

$$Senkung = Face_y - leftEye_y - \frac{Face_{width}}{4}$$
$$Senkung = Face_y - rightEye_y - \frac{Face_{width}}{4}$$
$$Senkung = Face_y - Mouth_y + \frac{Face_{width}}{4}$$

4.3. Darstellung des 3D-Modells mit Papervision3D

Das 3D-Modell einer Brille soll mit *Papervision3D* dargestellt werden. Bei der Initialisierung der *Papervision3D*-Szene ist zu beachten, dass die Breite und Höhe der Szene der des dargestellten Videobildes entspricht, da sonst eine korrekte Positionierung anhand der Trackingergebnisse erschwert wird. Damit eine 50x50 Pixel große Fläche aus *Papervision3D* auch auf der *Flash*-Bühne in dieser Größe dargestellt wird, muss die Eigenschaften des *Camera3D*-Objektes *Zoom = 11* und *Focus = 100* gesetzt werden. Außerdem ist zu beachten, dass sich der Koordinatenursprung bei *Papervision3D* in der Mitte der Bühne befindet, während in *Flash* der Koordinatenursprung in der oberen linken Ecke der Bühne liegt.

4.3.1. Positionierung anhand der Tracking-Ergebnisse

Nachfolgend sind die Formeln für die Berechnung der Position, Drehung, Neigung und Senkung des 3D-Modells aufgeführt. Die Faktoren wurden durch Tests ermittelt und sind dadurch spezifisch für eine Umsetzung mit *Papervision3D*. Bei der Verwendung einer anderen 3D-Engine können diese Faktoren abweichen.

$$rotationX = Senkung * 0.1$$
$$rotationY = (Drehung - Neigung) * 0.7$$
$$rotationZ = Neigung * 0.6$$

Die Position des 3D-Modells ist direkt von der Position des Gesichts abgängig und wird zusätzlich von den Rotationswinkeln beeinflusst. Für die Berechnung der y-Position gibt es wieder mehrere Möglichkeiten. $Stage_{width}$ und $Stage_{height}$ entspricht hier der Breite und Höhe der Bühne bzw. des Videobildes und $rotationX_{abs}$ dem Absolutwert von $rotationX$.

$$x = Face_x - Stage_{width} * 0.485 - rotationY * 2 - rotationZ * 0.6$$
$$y = Stage_{width} * 0.48 - leftEye_y - \frac{Face_{width}}{5} + Neigung * 0.6 + Senkung$$
$$y = Stage_{width} * 0.48 - rightEye_y - \frac{Face_{width}}{5} + Neigung * 0.6 + Senkung$$
$$y = Stage_{width} * 0.48 - Mouth_y - \frac{Face_{width}}{5} + Neigung * 0.6 + Senkung$$
$$z = rotationY - 1,5 * rotationX_{abs}$$

4.3.2. Realisierung der Verdeckung des 3D-Modells durch den Kopf

Die Verdeckung des 3D-Modells durch den Kopf soll durch ein vorkonstruiertes 3D-Modell erreicht werden. (siehe 3.4) Als Annäherung an die Form des Kopfes wird ein 3D-Modell in Form eines Zylinders erstellt und an die Position des Kopfes im Videobild gesetzt. Für die Nase wird ein Kegel im mittleren Drittel *Kopf*-Zylinders positioniert und leicht geneigt. (Abb. 24) Wie das 3D-Modell der Brille folgt dieser Zylinder den Bewegungen des Kopfes und verdeckt dabei Teile der Brille, die in der Realität vom Kopf selbst oder von der Nase verdeckt würden. Damit in der späteren Anwendung dieses *Kopf*-Modell nicht zu sehen ist, wird es mit 100% Grün texturiert. Anschließend wird dem *Viewport*-Objekt ein *Pixel Bender* Filter zugewiesen, der jeden Pixel mit einem Farbwert von 100% Grün transparent setzt.

4.4. Probleme und weiterführende Überlegungen

Bei ersten Tests der konzipierten Anwendung erwies sich vor allem das Tracking von Augen und Mund als fehleranfällig. Bei einem Verlust oder einer falsch erkannten Augen- oder Mundpartie wird die Position und Rotation des 3D-Modells nicht mehr korrekt dargestellt. Dies geschieht vorrangig bei großen Dreh-, Senk- und Neigungswinkeln des Kopfes sowie bei schlechten oder sich ändernden Lichtverhältnissen. Die Probleme des Trackings bei sich ändernden Lichtverhältnissen könnte durch eine automatische Helligkeitskorrektur gelöst werden. Der CAMSHIFT-Algorithmus erwies sich als weniger empfindlich gegenüber diesen Einflüssen.

Ein weiteres Problem ist die Performance der Anwendung. Auf dem Testrechner (siehe 3.3.1) konnte eine Bildwiederholrate von 10 Bildern pro Sekunde bei einer Berechnungszeit von 100 Millisekunden pro Bild erreicht werden. Die Bibliothek *FaceIt* verwendet für die Berechnungen ausschließlich die *Array*-Klasse. Zur Optimierung der Performance sollte statt dessen die *Vector*-Klasse verwendet werden. *Vectoren* können wesentlich schneller durchlaufen werden als *Arrays*.[119]

Zukünftig ist eine automatische Initialisierung des CAMSHIFT-Trackers denkbar. Dazu könnte in einem Initialisierungsprozess das Gesicht im Bild mittels Viola-Jones-Algorithmus gesucht werden und das Farb-Histogramm des gefundenen Bereich als Referenz-Histogramm für den CAMSHIFT-Tracker genutzt werden.

[119] vgl. Komponenten-Referenzhandbuch für ActionScript 3.0. Vector. `http://help.adobe.com/de_DE/AS3LCR/Flash_10.0/Vector.html`, letzter Aufruf:07.08.2010

5. Schlussfolgerung

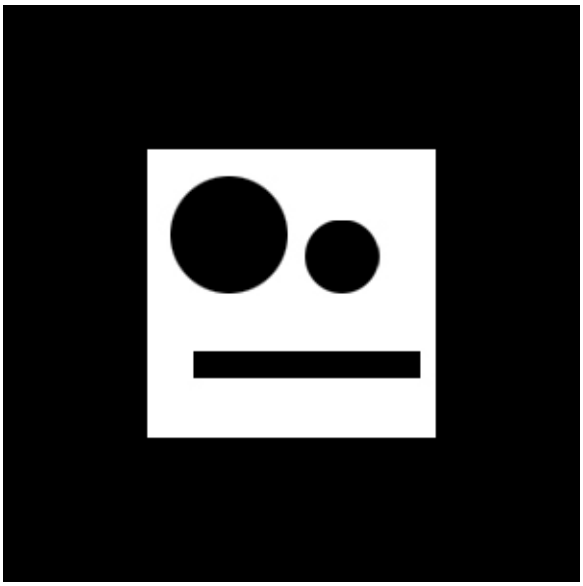
Ziel der Arbeit war es, die Möglichkeiten der Umsetzung von Augmented Reality in RIAs am Beispiel von *Adobe Flash* aufzuzeigen. Dafür wurden verschiedene Systeme für die markerbasierte Registrierung vorgestellt sowie die Vor- und Nachteile aufgeführt. Weiter wurde als Grundlage für eine markerlose Registrierung auf verschiedene Trackingbibliotheken für *Flash* eingegangen. Für die Darstellung von 3D-Modellen in *Flash* wurde eine Auswahl an 3D-Engines analysiert und verglichen. Im letzten Abschnitt wurden noch einmal die Probleme bei der Registrierung in markerlosen Augmented Reality Anwendungen deutlich. Auch wenn im Rahmen der Arbeit jeweils nur auf ausgewählte Verfahren und Bibliotheken eingegangen wurde, konnte die Zielsetzung der Arbeit dennoch erfüllt werden.

Das Verschmelzen von Realität und Virtualität wird nicht nur in RIAs in Zukunft eine immer größere Rolle spielen. Die denkbaren Anwendungsmöglichkeiten werden dabei nur von den technischen Möglichkeiten begrenzt. Augmented Reality profitiert dabei von der sich immer schneller entwickelnden Computerindustrie. Durch immer schnellere und kleinere Computer werden immer realistischere Darstellungen bis hin zu einer nahtlosen Verschmelzung von Realität und Virtualität möglich sein. Sowohl der Endverbraucher als auch die Industrie und Wirtschaft wird dabei von dieser Entwicklung profitieren.

Anhang

Anlagenverzeichnis

A. Pattern-Marker für Beispielanwendungen



B. Pixel Bender Filter für die Verdeckung vor statischem Hintergrund

```
<languageVersion : 1.0;>
kernel NewFilter

<   namespace : "Thomas Hurtig";
    vendor : "Your Vendor";
    version : 1;
    description : "Background Subtraction";
>
{
    input image4 video; //Variable video vom Typ image4 -> Videobild
    input image4 bg;    //Variable bg vom Typ image4 -> Hintergrundbild
    output pixel4 dst;  //Variable dst vom Typ pixel4 -> Ausgabe des Filters

    parameter float hT< //Variable hT vom Typ float -> Schwellwert Hue
        minValue:0.0;
        maxValue:0.5;
        defaultValue:0.04;
    >;

    parameter float sT< //Variable sT vom Typ float -> Schwellwert Saturation
        minValue:0.0;
        maxValue:0.5;
        defaultValue:0.04;
    >;

    //Funktion evaluatePixel wird fuer jeden Pixel durchlaufen
    void
    evaluatePixel()
    {
        //aktueller Pixel des Eingangsbild und des Hintergrundbildes
        //Rot, Gruen entsprechen Hue und Saturation
        float4 hsvBg =  sampleNearest(bg,outCoord());
        float4 hsvVid = sampleNearest(video,outCoord());

        //maskieren, wenn Absolutwert der Differenz zwischen den Hue
        //und Saturation Wert von Video- und Hintergrundbild
        //sonst durchsichtig
        if(abs(hsvBg.r - hsvVid.r)<=hT && abs(hsvBg.g-hsvVid.g)<=sT){
            dst = float4(1.0,1.0,1.0,1.0);
        }else{
            dst = float4(0.0,0.0,0.0,0.0);
        }
    }
}
```

C. Pixel Bender Filter für das Keying von 100% Grün

```
<languageVersion : 1.0;>

kernel ChromaKey
<    namespace : "Thomas Hurtig";
     vendor : "Your Vendor";
     version : 1;
     description : "Chroma Keying 0x00ff00";
>
{
     input image4 src; //Variable src vom Typ image4 -> Eingangsbild
     output pixel4 dst; //Variable dst vom Typ pixel4 -> Ausgabe des Filters

     //Funktion evaluatePixel wird fuer jeden Pixel durchlaufen
     void
     evaluatePixel()
     {
          //aktueller Pixel des Eingangsbild mit
          //Werten Rot, Gruen, Blau, Alpha
          pixel4 sample = sampleNearest(src,outCoord());

          //wenn Rot<=5 und Blau<=5 und Gruen>=250 werden alle Werte
          //des Pixels auf 0 gesetzt
          if(sample.r <= 0.02 && sample.g >= 0.98 && sample.b <= 0.02){
               sample = pixel4(0.0,0.0,0.0,0.0);
          }

          dst = sample;
     }
}
```

Literaturverzeichnis

Eigenständig erschienene Werke

[1] Abawi, Daniel F.: Analyse und Bewertung von Erstellungssystemen für Augmented Reality-Anwendungen. In: Frankfurter Informatik-Berichte; 2005, 4, Frankfurt am Main 2005 a

[2] Abawi, Daniel F.: Die Verdeckungsdarstellungsproblematik bei Augmented Reality-Anwendungen. In: Frankfurter Informatik-Berichte; 2005, 5, Frankfurt am Main 2005 b

[3] Azuma, Ronald T.: A Survey of Augmented Reality. In: Teleoperators and Virtual Environments 6, 4 (August 1997), 355-385.
`http://www.cs.unc.edu/~azuma/ARpresence.pdf`

[4] Bay, Herbert/Ess, Andreas/Tuytelaars, Tinne/Van Gool, Luc: SURF: Speeded Up Robust Features. In: Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp.: 346-359, 2008
`ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf`

[5] Bimber, Oliver/Raskar, Ramesh: Spatial Augmented Reality. Merging Real and Virtual Worlds. USA 2005

[6] Breen, David/Whitaker, Ross T./Rose, Eric/Tuceryaan, Mihran: Interactive Occlusion and Automatic Object Placement for Augmented Reality. In: Computer Graphics Forum, 15(3):11-22, 1996

[7] Cawood, Stephen/Fiala Ph.D., Mark: Augmented Reality. A Practical Guide. USA 2008

[8] Drascic, David/Milgram, Paul: Perceptual Issues in Augmented Reality. In: Proc. SPIE Vol. 2653: Stereoscopic Displays and Virtual Reality Systems III. California 1996
`http://etclab.mie.utoronto.ca/publication/1996/`

Drascic_Milgram_SPIE1996.pdf

[9] Fischer, Jan/Regenbrecht, Holger/Baratoff, Gregory: Detecting Dynamic Occlusion in front of Static Backgrounds for AR Scenes. In: Proceedings of the Workshop on Virtual Environments 2003, Seiten 153-161. ACM Press, 2003

[10] Friedrich, Wolfgang (Hrsg.): ARVIKA. Augmented Reality für Entwicklung, Produktion und Service. Bonn/Berlin 2004

[11] Kato H./Billinghurst M.: Marker tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In: Augmented Reality, (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on, pp.: 85-94, 1999

[12] McGarrity, Erin/Tuceryan, Mihran: A Method for Calibrating See-Through Head-Mounted Displays for AR Augmented Reality. In: (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on, pp.: 75-84, 1999

[13] Milgram, Paul/Colquhoun Jr., Herman: A Taxonomy of Real and Virtual World Display Integration. In: Ohta, Yuichi/Tamura, Hideyuki(Hrsg.): Mixed Reality. Merging Real and Virtual Worlds. Japan/North America/Deutschland 1999

[14] Milgram, Paul/Takemura, Haruo/Utsumi, Akira et al.: Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. In: Proc. SPIE Vol. 2351, Telemanipulator and Telepresence Technologies. 1994
http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf

[15] Nakamae, Eihachiro/Harada, Koichi/Ishizaki, Takao/Nishita, Tomoyuki: A Montage Method: The Overlaying of the Computer Generated Images onto a Background Photograph. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, Seiten 207-214. ACM Press, 1986

[16] Sutheland, Iva E.: A Head-Mounted Three Dimensional Display. In: AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I. USA 1968

[17] Teichrieb, Veronica/do Monte Lima, João Paulo Silvia/Apolinário, Eduardo Lourenço/de Farias, Thiago Souto Maior Cordeiro/Bueno, Márcio Augusto Silva/Kelner, Judith/Santos, Ismael H. F.: A Survey of Online Monocular Markerless Augmented Reality. In: International Journal of Modeling and Simulation for the Petroleum Industry, VOL. 1, NO. 1. Rio de Janeiro August 2007

[18] Viola, Paul/Jones, Michael: Robust Real-time Object Detection. In: Second International Workshop on Statistical and Computional Therories of Vision-Modeling, Learning, Computing, and Sampling. Canada 2001 `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.4987&rep=rep1&type=pdf`

Internetquellen

[19] ActionScript 3.0 Language and Components Reference. Display Object. `http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/DisplayObject.html#mask`, letzter Aufruf:29.06.2010

[20] ActionScript 3.0 Language and Components Reference. ColorMatrixFilter. `http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/filters/ColorMatrixFilter.html`, letzter Aufruf: 29.06.2010

[21] Augmented Driving. `http://www.imaginyze.com/Site/Welcome.html`, letzter Aufruf: 08.06.2010

[22] Augmented Driving Screenshots. `http://www.imaginyze.com/Site/Screenshots.html#4`, letzter Aufruf: 08.06.2010

[23] Augmented Reality for Flash. flare*nft. `http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Anft`,

letzter Aufruf: 03.06.2010)

[24] Augmented Reality für Flash. flare*tracker. http://www.imagination.at/?Produkte:Augmented_Reality_f%FCr_Flash:flare%2Atracker#, letzter Aufruf: 03.06.2010

[25] Away3D FP10 V3.5.0 API Documentation. http://away3d.com/livedocs/3.5.0_lib/, letzter Aufruf: 21.06.2010

[26] Away3DLite API Reference. http://away3d.com/livedocs/fp10/Away3DLite/, letzter Aufruf:21.06.2010

[27] Brimelow, Lee: Augmented Reality with FLARManager. Learn how to use FLARManager library to make creating AR applications a breeze. http://www.gotoandlearn.com/play.php?id=114, letzter Aufruf: 02.06.2010

[28] Brimelow, Lee: Introduction to Augmented Reality. Learn the basics of creating an AR application using the FLARToolKit. http://www.gotoandlearn.com/play.php?id=105, letzter Aufruf: 02.06.2010

[29] Camera Calibration. http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm, letzter Aufruf: 07.06.2010

[30] CASTING SHADOWS IN PAPERVISION - REDUX.http://blog.zupko.info/?p=146, letzter Aufruf: 21.06.2010

[31] Castrillón-Santana, Modesto: Face Features Detectors ftp://mozart.dis.ulpgc.es/pub/Software/HaarClassifiers/FaceFeaturesDetectors.zip, letzter Aufruf:03.08.2010

[32] COLLADA. https://collada.org/mediawiki/index.php/COLLADA, letzter Aufruf: 21.06.2010

[33] Computer-Vision-Algorithm. http://www.hitl.washington.edu/artoolkit/documentation/vision.htm, letzter Aufruf: 12.05.2010

[34] Computer Vision Face Tracking For Use in a Perceptual User Interface. http://isa.umh.es/pfc/rmvision/opencvdocs/papers/camshift.pdf, letzter Aufruf: 07.07.2010

[35] Creating Roadmaps for Surgeons. Using the Mac for Better Surgical Navigation. http://www.apple.com/science/profiles/maki/, letzter Aufruf: 09.06.2010

[36] For Nerds Only: Custom FLAR Markers Explained. http://www.squidder.com/2009/03/05/for-nerds-only-custom-flar-markers-explained/, letzter Aufruf: 8.6.2010

[37] GPS-Global Positioning System. http://www.elektronik-kompendium.de/sites/kom/1201071.htm, letzter Aufruf: 15.06.2010

[38] Haar, Adam: OpenCV Face Detection Visualized. http://vimeo.com/12774628, letzter Aufruf: 08.07.2010

[39] Hameed, Shameem: HAARcascade for eyes. http://www-personal.umich.edu/~shameem/haarcascade_eye.html, letzter Aufruf:03.08.2010

[40] How does ARToolKit work? http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm, letzter Aufruf: 14.05.2010

[41] How to make only "The Hole" using Papervision3D. http://saqoosha.net/en/2009/01/08/1676/, letzter Aufruf: 29.06.2010

[42] How to Use SimpleShadow. http://away3d.com/tutorials/how-to-use-simpleshadow, letzter Aufruf: 21.06.2010

[43] IKEA | BILLY 30 år. http://www.ikea.com/ms/sv_SE/kampanj/fy10/billy30/, letzter Aufruf: 02.06.2010)

[44] Jensen, A.: Laparoskopie-Zentrum Bochum. http://www.frauenklinik-uni-bochum.de/pdf/lsk_zentrum.pdf, letzter Aufruf: 08.06.2010

[45] Jung, B: Camshift: going to the source. http://www.mukimuki.fr/flashblog/2009/06/18/camshift-going-to-the-source/, letzter Aufruf: 07.07.2010

[46] Marilena. http://www.libspark.org/wiki/mash/Marilena, letzter Aufruf: 08.07.2010

[47] Marker"s" Generator Online Released! http://flash.tarotaro.org/blog/2009/07/12/mgo2/, letzter Aufruf: 08.06.2010

[48] MINI II. http://www.flickr.com/photos/metaio/4473598564/, letzter Aufruf: 02.06.2010

[49] Minimalinvasive Laparoskopie mit OsiriX. http://aycandigital.blogspot.com/2009/02/minimalinvasive-laparoskopie-mit-osirix.html, letzter Aufruf: 09.06.2010

[50] Moment (Bildverarbeitung). http://de.wikipedia.org/wiki/Moment_(Bildverarbeitung), letzter Aufruf: 07.07.2010

[51] Normal Mapping. http://de.wikipedia.org/wiki/Normal_Mapping, letzter Aufruf: 21.06.2010

[52] Papervision3D API Documentatio. http://www.papervision3d.org/docs/as3/index.html, letzter Aufruf: 21.06.2010

[53] Pixel Bender Technology Center. http://www.adobe.com/devnet/pixelbender/, letzter Aufruf: 29.06.2010

[54] Ray Ban. Virtual Mirror. http://www.ray-ban.com/germany/science/virtual-mirror, letzter Aufruf: 14.07.2010

[55] Rich Internet Application Statistics. Real World Stats of RIA Plugin Developments. http://riastats.com/, letzter Aufruf: 04.06.2010

[56] Rich Internet Applications Market Share. RIA Market Penetration and Global Usage. http://www.statowl.com/custom_ria_market_penetration.php?1=1&timeframe=ytd&interval=month&chart_id=13&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&timeframe=last_6, letzter Aufruf; 04.06.2010

[57] Sandy 3D Engine, 3.1.2 http://sandy.googlecode.com/svn/trunk/sandy/as3/tags/3.1.2/docs/index.html, letzter Aufruf: 21.06.2010

[58] Schmidt, Michael/Schmitz Esther: Gaming Sickness. http://user.cs.tu-berlin.de/~edda/gamingsickness.html, letzter Aufruf: 04.05.2010

[59] Socolofsky, Eric: FLARManager. Augmented Reality in Flash. http://words.transmote.com/wp/flarmanager/, letzter Aufruf: 08.06.2010

[60] Socolofsky, Eric: FLARManager for flare/NFT. http://words.transmote.com/wp/20100520/flarmanager-for-flarenft/, letzter Aufruf: 08.07.2010

[61] Studierstube Tracker. http://studierstube.icg.tu-graz.ac.at/handheld_ar/stbtracker.php, letzter Aufruf: 12.05.2010

[62] The Lego Group to Boost Retail with Metaio. Pressemittelung der metaio GmbH, Dezember 2008, http://www.metaio.com/fileadmin/homepage/Presse/Dokumente/Pressemitteilungen/English/E_2009_01_15_PR_LEGO.pdf, letzter Aufruf: 09.06.2010

[63] Total Immersion. Datasheet @Home Applications. http://www.t-immersion.com/lc-content/uploads/2009/11/datasheethome.pdf, letzter Aufruf: 05.06.2010

[64] Total Immersion. FAQ. http://www.t-immersion.com/en,faq,825.html, letzter Aufruf: 05.06.2010

[65] USPS Priority Mail - Virtual Box Simulator. https://www.prioritymail.com/simulator.asp, letzter Aufruf: 02.06.2010

[66] What is FLARToolKit. http://www.libspark.org/wiki/saqoosha/FLARToolKit/en, letzter Aufruf: 07.06.2010

[67] Wikitude Drive: Never Take Your Eyes of the Road Again. http://www.wikitude.org/drive-2, letzter Aufruf: 08.06.2010

[68] World Premiere: The Augmented Reality Ad For MINI. http://www.metaio.de/fileadmin/homepage/Dokumente/English/Best_Practice_MINI.pdf, letzter Aufruf: 02.06.2010

[69] Zatepyakin, Eugene: ASSURF. SURF library for Adobe Flash Platform. http://code.google.com/p/in-spirit/wiki/ASSURF, letzter Aufruf: 08.07.2010

[70] Zatepyakin, Eugene: Haar Cascade Detector. http://code.google.com/p/in-spirit/wiki/HaarCascadesDetector, letzter Aufruf: 03.08.2010

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage dem Prüfungsausschuss des Fachbereiches Medien eingereichte Bachelorarbeit zum Thema

„Umsetzung von Augmented Reality in Rich Internet Applications am Beispiel von Adobe Flash“

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Ort, Datum Unterschrift